Universidad Nacional Mayor de San Marcos

Universidad del Perú. Decana de América

Facultad de Ingeniería de Sistemas e Informática

Escuela Académico Profesional de Ingeniería de Sistemas

Solución distribuida orientada a servicios para la transferencia de datos entre agentes del sistema de EPS a través del uso de servicios web

TESINA

Para optar el Título Profesional de Ingeniero de Sistemas

AUTORES

Juan Jesús ORBEZO CHUCHÓN

Víctor Omar POLO TEJEDA

ASESOR

Rubén Alexander GIL CALVO

Lima, Perú

2008

Referencia bibliográfica

Orbezo, J. & Polo, V. (2008). *Solución distribuida orientada a servicios para la transferencia de datos entre agentes del sistema de EPS a través del uso de servicios web*. Tesis para optar el título profesional de Ingeniero de Sistemas. Escuela Académico Profesional de Ingeniería de Sistemas, Facultad de Ingeniería de Sistemas e Informática, Universidad Nacional Mayor de San Marcos, Lima, Perú.

DEDICATORIA: *Dedicamos el presente trabajo a nuestros padres y hermanos por el apoyo constante que nos brindan y con quienes compartimos los mejores momentos de nuestras vidas.*

RESUMEN

SOLUCION DISTRIBUIDA ORIENTADA A SERVICIOS PARA LA TRANSFERENCIA DE DATOS ENTRE AGENTES DEL SISTEMA DE EPS A TRAVES DEL USO DE SERVICIOS WEB

JUAN JESÚS ORBEZO CHUCHÓN

VÍCTOR OMAR POLO TEJEDA

JUNIO - 2008

Asesor : Rubén Gil Calvo

Título : Licenciado en Computación

El reciente crecimiento en la cantidad de afiliados en el mercado de EPS, ha ocasionado una mayor demanda de Sistemas de Información capaces de solucionar las diversas necesidades de los diferentes agentes del Sector EPS y que se establezcan como integradores de los procesos de negocios de dichos agentes.

Por tal razón, la presente tesina pretende dar una solución acorde a los requerimientos exigidos en el Sistema de EPS, teniendo como pilares a los Servicios Web, debido a que brindan a las organizaciones puentes de comunicación entre el software escrito en lenguajes diferentes, desarrollados

por diferentes fabricantes e incluso que trabajan sobre diferentes sistemas operativos, asegurando así la interoperabilidad entre aplicaciones heterogéneas.

Palabras Claves:

Servicios Web

SEPS

EPS

Interoperabilidad

XML

ABSTRACT

DISTRIBUTED SERVICE-ORIENTED SOLUTION FOR INFORMATION TRANSFER AMONG EPS SYSTEM AGENTS ACROSS WEB SERVICES

JUAN JESÚS ORBEZO CHUCHÓN

VÍCTOR OMAR POLO TEJEDA

JUNE - 2008

Adviser	:	Rubén Gil Calvo
University Degree	:	Licentiate in Computing

The recent growth in the quantity of members on EPS market, has caused a major demand of Information Systems capable of solving the diverse needs of the different agents of the Sector EPS and that are established as integrators of the business processes of the above mentioned agents.

For such a reason, the present dissertation tries to give a solution chord to the requirements demanded in EPS System, taking Web Services as pillars, due to the fact that bridges of communication offer to the organizations between the software written in different languages, developed by different manufacturers and even that work on different operating systems, assuring this way the interoperability between heterogeneous applications.

Keywords:

Web Services

SEPS

EPS

Interoperability

XML

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPITULO I

INTRODUCCIÓN

Este capitulo describe el panorama general en que se sitúa el problema que se aborda en esta investigación y para el cual se plantea una solución. Se presentan los objetivos que se alcanzaron durante el desarrollo de la misma, así como la metodología que se siguió. Finalmente, se mencionan las contribuciones del trabajo y la estructura del presente documento.

1.1. Antecedentes del Problema

El 17 de Mayo de 1997, mediante la Ley Nº 26790, Ley de Modernización de la Seguridad Social en Salud, se creó la Superintendencia de Entidades Prestadoras de Salud – SEPS, como Organismo Público Descentralizado del Sector Salud, con las funciones de autorizar, regular y supervisar el funcionamiento de las Entidades Prestadoras de Salud – EPS, cautelando el uso correcto de los fondos que administran y el cumplimiento de las normas legales y reglamentarias correspondientes, en resguardo de los derechos de los asegurados.

En pos del cumplimiento de las mencionadas funciones, la SEPS desarrolló e implantó el SITEDS (Sistema Integrado de Transacciones Electrónicas de Datos en Salud), el cual consiste en una red abierta de

comunicación electrónica entre los agentes participantes del Sistema de EPS[1] y Seguros de Salud, con capacidad para ofrecer servicios de transacciones y consulta de información clínica, administrativa y financiera hasta nivel de paciente, manteniendo las condiciones de seguridad pertinentes.

Desde el año 2006, se ha presentado un crecimiento del uso de los módulos de consulta y autorización de los servicios de salud y una mayor demanda de soporte (actualmente implementado en 349 entidades vinculadas al sistema y 3 de las 4 EPS existentes).

El lanzamiento de las primeras versiones del SITEDS, trajo consigo la necesidad de que las clínicas que cuentan con sistemas de desarrollo propios, es decir, con soluciones integradas para sus procesos internos, tengan la imperiosa necesidad de contar con la integración de las funcionalidades del SITEDS en sus sistemas propios.

La solución SITEDS basada en componentes Microsoft COM (Component Object Model) cubrieron alguna de las necesidades de las clínicas que tenían sistemas que eran compatibles con esta tecnología y por ende podían tener interoperabilidad con los componentes COM – del tipo dll (dynamic linking library) - transaccionales del SITEDS. Un número reducido de estas clínicas que no podían interpretar los componentes mencionados, tuvieron que manejarlos, hasta la actualidad, como sistemas aislados.

[1] En la actualidad los agentes del Sistema de EPS son Rímac EPS, Pacífico EPS, Mapfre EPS y Persalud EPS y las Entidades Vinculadas (clínicas, centros médicos de apoyo, etc.). Para más información ver Los agentes del Sistema en http://www.seps.gob.pe.

A medida que surgían nuevos requerimientos de los agentes participantes en el sistema de EPS, la SEPS contrataba los servicios de una consultora Informática, creadora esta de la solución, para la atención a dichos requerimientos. Pero a partir del año 2005 por el rompimiento de las relaciones de la SEPS con dicha consultora, se vio en la imperiosa necesidad de contar con alguien que conozca la lógica y el código fuente de dicha solución, por lo que partir de ese año en adelante hasta el 2006 se contrató a un ex-trabajador que había participado en el desarrolló de dicho aplicativo. Este profesional era contratado tanto por la SEPS como por las EPS para realizar los mantenimientos respectivos y atención a los requerimientos complejos del sistema. Con lo cual se generó una dependencia de la SEPS y de las EPS con este Profesional de Informática.

1.2 Definición del Problema

La infraestructura tecnológica actualmente empleada por los agentes participantes en el sistema de EPS, hace que se generen con cierta frecuencia nuevos requerimientos al Sistema, los cuales tienen que ser atendidos en el corto/mediano plazo, dependiendo de la complejidad, además en muchos casos su atención resulta ser costosa para la SEPS. Este problema es producto de la dependencia hacia un profesional de informática para la atención de los requerimientos solicitados al Sistema.

Otro problema es que el sistema carece parcialmente de interoperabilidad con sistemas heterogéneos[2], lo cual limita a que Entidades Vinculadas puedan integrar en sus sistemas los servicios del SITEDS, a esto también se suma la limitante de no transmitir documentos binarios mediante tecnología estándar.

1.3 Objetivos

1.3.1 Objetivo General

El objetivo de la presente investigación es diseñar y desarrollar una solución distribuida que permita realizar transferencias de datos entre los Agentes del Sistema de EPS -conservando la actual lógica de negocios- a través de Servicios Web con una política de seguridad y confidencialidad definida por la SEPS. Dicha solución será concebida como alternativa al software SITEDS, empleado actualmente por los agentes participantes en el sistema de EPS.

1.3.2 Objetivos Específicos

- Utilizar protocolos Web estándares para el tranporte de datos de la solución (HTTP, SOAP).
- Establecer como elemento de control y administración de toda la solución distribuida a la SEPS.
- Reutilizar la lógica de negocios actual del Sistema de EPS.
- Garantizar la seguridad y confidencialidad de los datos de la solución mediante el uso del protocolo estándar WS-Security.

[2] Un sistema heterogéneo es aquel que se encuentra compuesto por hardware con características físicas distintas entre sí, y software con características operativas distintas entre sí, pero que se pueden comunicar utilizando medios comunes.

- Alcanzar una modularidad y flexibilidad de la solución.
- Plantear un caso de estudio, así como su análisis y solución integrando los puntos anteriores.

1.4 Justificación

Los avances en la tecnología y el avanzado grado de desarrollo de los estándares de la industria que posibilitan una arquitectura orientada a servicios dan la posibilidad de alinear el negocio con la infraestructura tecnológica hasta niveles que difícilmente pueden lograrse con arquitecturas anteriores, y posibilitan la agilidad para responder ante nuevas demandas, aumentar la facilidad de adaptación, además de la posibilidad de disminuir costos de desarrollo al incrementar la reutilización [IBM 2006].

Con base en la premisa dada, en la selección del presente tema de investigación influyeron como factores determinantes la necesidad de desarrollar una tecnología basada en estándares y que tenga la capacidad de integrar los principales procesos de nogocios entre las EPS que se están sumando al Sistema y la SEPS, Entidad en la que nos desempeñamos como profesionales; otro factor es el deseo de aportar a esta institución nuestros conocimientos en este tema para expandirlos y obtener experiencia en nuestra formación como ingenieros de sistemas. Y Como factor personal podemos citar la especial afición que tenemos por las tecnologías distribuidas.

Este tema tiene una primordial relevancia en el Sistema de EPS porque muchas de las Entidades Vinculadas con desarrollo propio necesitan integrar en sus sistemas los métodos del proceso empleados por el SITEDS para la atención o prestación de salud a los asegurados de una determinada EPS para su control y administración interna, y esto es posible mediante la Arquitectura Orientada a Servicios.

Otro punto que hace que esta investigación cobre importancia es que estos Servicios a un futuro a corto o mediano plazo pueden tener un costo real en el mercado, es decir, la SEPS podría cobrar a cada Entidad Vinculada la utilización de los Servicios Web o la implementación de algunos nuevos, además que los costos de desarrollo y mantenimiento para la SEPS se reducirán considerablemente.

Además podemos agregar que esta solución permitirá la interoperabilidad con otras plataformas heterogéneas y la integración con softwares escritos en cualquier lenguaje de programación, así como también se aminorará el impacto de esta nueva solución informática en los usuarios finales, con la reutilización del componente cliente del SITEDS.

Para terminar podemos resaltar que la Arquitectura Orienteada a Servicios mediante Servicios Web acelerará el proceso de atención de los requerimientos de las EPS y reducirá su costo para la SEPS, que actualmente tiene dependencia hacia un consultor externo que fue parte del

equipo de profesionales de informática que realizaron la implementación del SITEDS en el sistema de EPS.

1.5 Alcances

Este trabajo de investigación podrá ser utilizado por cualquier EPS que solicita el servicio o desee migrar su sistema de transacciones actual a la arquitectura que se presenta en esta investigación. En este trabajo de investigación nos limitaremos a estudiar la solución aplicada a la Superintendecia de Entidades Prestadoras de Salud , como ente administrador del sistema , y a MAPFRE Perú EPS que es la solicitante del servicio, que ingreso al mercado de las EPS a fines del año 2007.

1.6 Propuesta

La adopción de arquitecturas SOA, basadas en Servicios Web. SOA es fundamentalmente, una metodología de desarrollo que fomenta compartir servicios de aplicación que pueden ser invocados de manera remota a través de las redes. Es una forma de hacer más con menos, donde las aplicaciones pueden ser construidas más rápidamente y, cada vez, con menos líneas de código original. Algunos de sus beneficios son cuantificables, pero el más importante reside en su capacidad de alineamiento con las necesidades de las organizaciones dinámicas y flexibles del siglo XXI.

Nuestra propuesta en la presente investigación es implementar una arquitectura SOA basada en Servicios Web como medio de comunicación entre los agentes participantes en el sistema de EPS, empleando mensajes XML y SOAP como protocolo de comunicación.

El sistema se basa fundamentalmente en el reemplazo de los componentes de Servidor EPS y SEPS, por Servicios Web, reutilizando la lógica de negocio de dichos componentes.

El Servicio Web EPS, expondrá las funcionalidades del negocio del sistema, pero conservara la estructura de data enviada, es decir, los datos expuestos seguirán manteniendo el formato de trama de datos concatenados[3] estandarizados por la SEPS.

El Servicio Web SEPS, será el encargado de administrar las transacciones realizadas por las EV[4], este tendrá como función:

- Direccionar o Enrutar al Servicio Web de determinada EPS consultada.
- Autentificar a los usuarios, como medio de seguridad y gestión de usuarios.
- Cifrar la trama de respuesta, para mantener la confidencialidad de la información.
- Gestionar y administrar el acceso a los Servicios que brinda el SITEDS.

[3] Ver Anexo 03 al final del documento.
[4] Entidad Vinculada.

En el componente Cliente desarrollado en Visual Basic 6.0, se implementará la funciones necesarias para consumir el Servicio Web SEPS, para poder realizar dicho consumo se emplearan una library assembly .NET, las cuales proveen capacidades de consumo de Servicios Web y de interoperabilidad con aplicaciones COM, siendo la razón del empleo estas librerías la existencia de protocolos estándares soportados por el Net Framework 2.0.

El desarrollo de la nueva solución informática será realizada en lenguaje Visual Basic 2005 sobre Framework .Net 2.0, y WSE 3.0 que es una implementación de diferentes especificaciones WS-*. Web Services Security [OAS 2006].

Se empleara el WS-Security como protocolo de comunicación estándar de seguridad para la autentificación y autorización de usuarios, así como también para el cifrado o encriptado del Mensaje sobre HTTP.

1.7 Organización de la Investigación

La presente investigación esta organizado en 7 capítulos y ha sido estructurada de la siguiente manera. En el capitulo 1 presentamos la introducción de la investigación en su conjunto. En el Capitulo 2 presentamos con detalle los fundamentos teóricos relacionadas con la propuesta de la investigación. En el Capitulo 3 presentamos el estudio profundo del problema, las Tecnologías que se emplean actualmente. Así como también la descripción del Caso de Estudio. En el Capitulo 4

presentamos la resolución del problema, existiendo solo una técnica seleccionada. En el capitulo 5 presentamos el sistema de computación para la resolución del problema, es decir, el análisis y diseño de la solución. En el Capitulo 7 presentamos el conjunto de elementos que permitió la identificación de la fuente documental de la que se extrae la información.

CAPITULO II

MARCO TEORICO

En este capítulo de la investigación examinamos los conceptos fundamentales acerca de sistemas distribuidos, la infraestructura que soporta la solución propuesta mediante Servicios Web. Comenzamos con una investigación de la evolución experimentada por los sistemas de computación y la motivación que provocó el paso de sistemas centralizados a distribuidos. Presentamos diversas definiciones de un sistema distribuido, identificando los elementos comunes que permiten caracterizarlo, sus ventajas y desventajas, así como algunos ejemplos muy conocidos. Luego, revisamos la arquitectura general de un sistema distribuido, basada inicialmente en el modelo *cliente-servidor* y posteriormente en el paradigma de *orientación a objetos*. Por último, abordamos el propósito de un *middleware*, las implementaciones más populares hoy en día, mencionando cinco estándares para desarrollar sistemas distribuidos, destacando ellos los Servicios Web y la Arquitectura Orientada a Servicios (SOA), modelo para el desarrollo de nuestra solución.

2.1. Sistemas Distribuidos

2.1.1. Evolución

En sus inicios, los sistemas de información se limitaban a programas mono-usuarios residentes en grandes computadoras. Luego,

aparecieron los *mainframes* de tiempo compartido que podían atender a varios usuarios desde terminales remotas para el acceso a recursos centralizados mediante comandos en texto normal. Con la llegada del *microprocesador* aparecen las computadoras personales (*PC's*) que, junto a las redes de área local (*LAN*, *Local Area Network*), permitieron la descentralización de los recursos. Con base en una *arquitectura cliente-servidor*, las aplicaciones se dividieron en partes que podían residir en computadoras distintas. El *boom* de *Internet*, la reducción en tamaño y costo del hardware informático, el aumento en la capacidad de las computadoras personales y el desarrollo de las tecnologías de red, dan como resultado *sistemas distribuidos* con *PC's* más poderosas conectadas por medio de redes de alta velocidad. Una aplicación distribuida puede ejecutar partes de su código en distintos procesadores que coordinan la ejecución mediante mensajes. En los últimos años, los sistemas distribuidos se han consolidado y son la base para el desarrollo de tecnologías como *grids*, *clusters*, la comunicación *peer to peer* y las redes de dispositivos móviles.

2.1.2. Causas del surgimiento de los Sistemas Distribuidos

La principal causa detrás de la consecución del paradigma distribuido, se debe a la necesidad de compartir recursos. En el mundo de la informática, el término *recurso* comprende dispositivos de hardware (por ejemplo procesador, disco duro, impresora) así como entidades de software (por ejemplo archivos, bases de datos, secuencias de imágenes, audio de un teléfono móvil) [CDK 2001]. En la actualidad,

compartir recursos por medio de una red de computadoras es una característica tan familiar que es aprovechada en diversos ámbitos tales como: los negocios, la investigación, el gobierno y el hogar [A&W 2004].

2.2. Bases Teóricas

2.2.1. Definición

Es difícil capturar en una única definición las diversas facetas que presentan los sistemas distribuidos [V&R 2001]. Se enuncian a continuación varias de las definiciones emitidas por diversos autores:

"Un sistema distribuido es una colección de computadoras autónomas que trabajan conjuntamente para dar la apariencia de un sistema coherente único." [T&V 2002]

"Un sistema distribuido es aquel en el que los componentes de hardware o software, localizados en computadoras conectadas mediante red, comunican y coordinan sus acciones sólo mediante paso de mensajes." [CDK 2001]

"Un sistema distribuido es una colección de dispositivos individuales de computación que pueden comunicarse unos con otros." [A&W 2004]

"Un sistema distribuido es un sistema compuesto de varias computadoras que se comunican mediante una red, almacenando

procesos que utilizan un conjunto de protocolos distribuidos común para soportar la ejecución coherente de actividades distribuidas." [V&R 2001]

"Un sistema distribuido es un sistema de procesamiento de información que contiene múltiples computadoras independientes que cooperan unas con otras sobre una red de comunicaciones para alcanzar un objetivo específico." [PRP 2006]

Tal gama de definiciones obliga a identificar ciertos elementos en común que, en conjunto, permitirán caracterizar de mejor manera un sistema distribuido, así se tienen:

- Varias computadoras o procesos, denominados *nodos*.
- Red de comunicaciones o de computadoras, denominada simplemente red.
- Intercambio de mensajes (coordinación).
- Meta común, por ejemplo compartir un recurso, brindar un servicio, o ejecutar una aplicación.
- Apariencia de un sistema único.

Cada una de las características mencionadas, por sí solas, no definen un sistema distribuido, por ejemplo una red de computadoras no es un sistema distribuido ya que las computadoras en una red quizás nunca interactúen, o están limitadas a recibir o enviar mensajes de manera ocasional [V&R 2001].

Una diferencia importante es que los procesos en un sistema distribuido cooperan para alcanzar un objetivo común. Cabe recalcar que, desde el punto de vista de los usuarios, el conjunto de recursos disponibles en un sistema distribuido actúa como un único sistema virtual [KSSRM 2002].

2.2.2. Ejemplos de Sistemas Distribuidos

Los enormes progresos tanto en tecnologías de información como en redes de comunicaciones han guiado hacia ambientes de computación distribuida con una multitud de servicios [G&G 1996]. En este contexto, *Internet* constituye una gran infraestructura para la construcción de sistemas distribuidos.

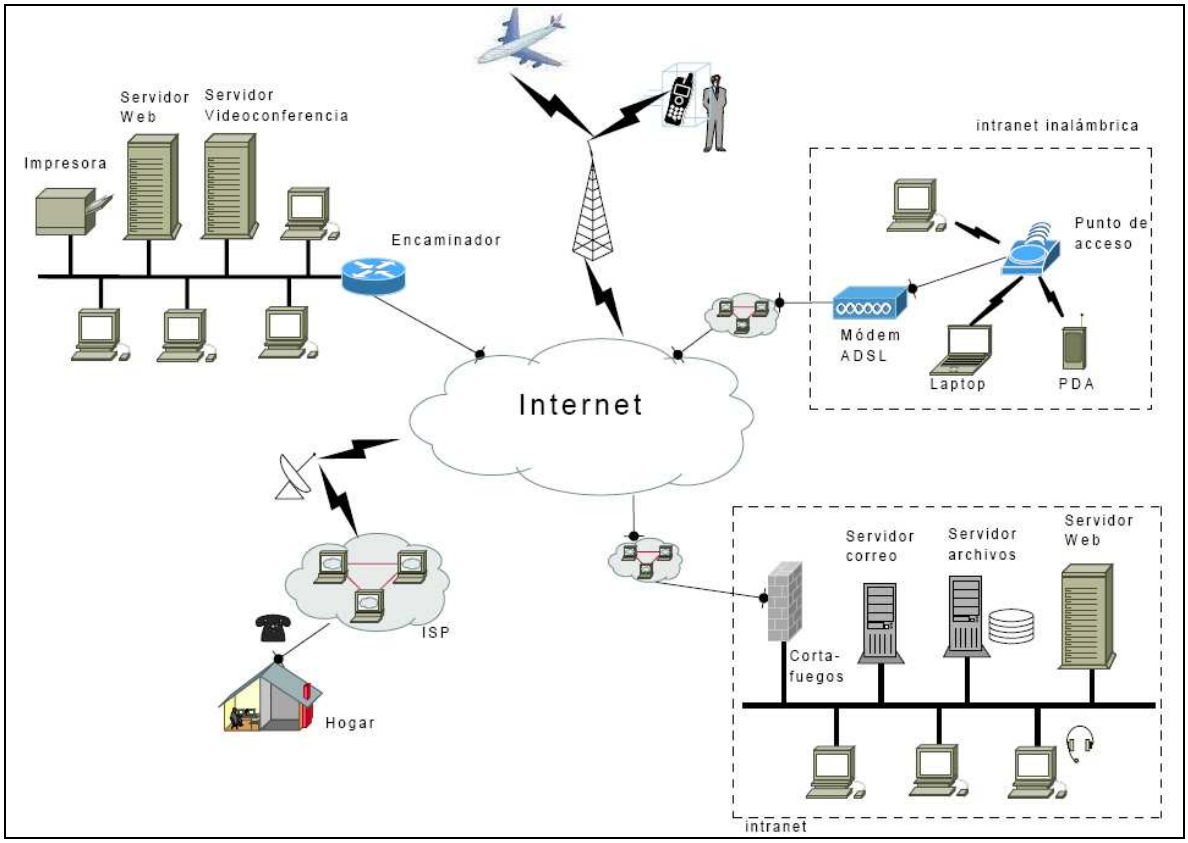

Figura 2.1. Ejemplo de Sistemas Distribuidos [MEJ 2007]

En la figura 2.1 se presentan varios ejemplos de este tipo de sistemas, *por ejemplo* un servidor *Web* ubicado en la *intranet* de una organización (red privada con tecnología *Internet*) proporciona servicio tanto a usuarios que pueden estar dentro de la organización, fuera de ella o incluso desde el hogar a través de empresas que proveen acceso a *Internet* (*ISP*, *Internet Service Provider*). De la misma forma, servicios como el correo electrónico, transferencia de archivos, multimedia e impresión, están disponibles para usuarios internos y externos. Los avances en cuanto a tecnología inalámbrica han permitido que dispositivos como computadoras portátiles (*Laptops*), *PDA's* (*Personal Digital Assistant*), teléfonos celulares y, hasta dispositivos integrados en aviones y automóviles, sean capaces de conectar a usuarios mientras ellos están en movimiento.

2.3. Ventajas y desventajas de los Sistemas Distribuidos

El paso de sistemas centralizados a distribuidos ha sido el resultado de un proceso de evolución, guiado principalmente por la mentalidad de que compartir es mejor que duplicar. A través de este proceso, han surgido nuevas posibilidades además de poder compartir recursos tanto físicos como lógicos. Podemos mencionar el acceso a computadoras distantes geográficamente, la ejecución de código en computadoras remotas, la integración de varias computadoras con el fin de aumentar la capacidad de procesamiento y así poder resolver problemas complejos, *etc*.

La tabla 2.1 presenta una comparación entre sistemas distribuidos y sistemas centralizados, tomando en consideración diversos criterios que son importantes para decidir la utilización de una u otra tecnología. Se ha reunido la información encontrada en [V&R 2001], [PRP 2006], y [TAN 2003]:

Criterio	*Sistemas centralizados*	*Sistemas distribuidos*
Acceso a recursos	Local	Global
Tecnología de los nodos	Homogénea	Heterogénea
Administración	Una organización (simple)	Varias (compleja)
Coste económico	Bajo	Alto
Modularidad	Un nodo	Varios nodos
Complejidad	Baja	Alta
Consistencia	Simple	Compleja
Escalabilidad	Restringida	Ilimitada
Seguridad	Alta	Baja
Rendimiento/Velocidad	Bajo	Alto
Riesgo de fallos	Bajo	Alto
Disponibilidad	Baja	Alta
Tolerancia a fallos	Baja	Alta

Tabla 2.1. Comparación entre Sistemas Centralizados y Sistemas Distribuidos

Podemos apreciar que los sistemas distribuidos ofrecen una serie de ventajas muy importantes, sin embargo, no es menos cierto que éstas son contrarrestadas por algunas desventajas. Por tanto, no siempre una solución distribuida es la más adecuada, solamente cuando la naturaleza del problema así lo determina. De tal forma, cada aplicación requiere de un análisis exhaustivo para concluir si su desarrollo, bajo el enfoque distribuido, resulta beneficioso.

2.4. Arquitectura general de un Sistema Distribuido

Corresponde a la estructura lógica y física de los elementos que intervienen en un sistema distribuido así como la manera de interactuar entre ellos.

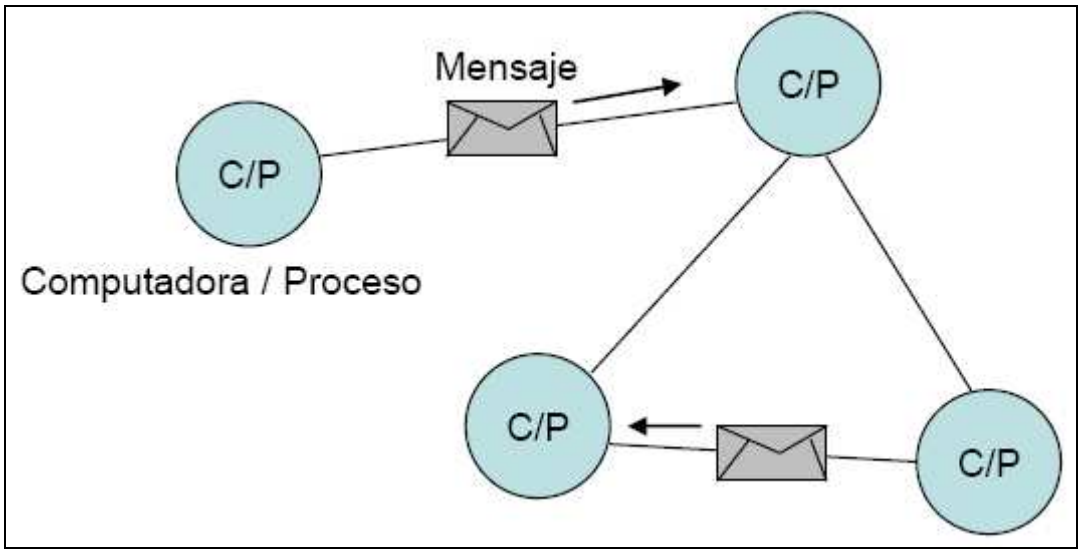

Figura 2.2. Arquitectura general de un Sistema Distribuido [MEJ 2007]

En la figura 2.2 se pueden establecer dos perspectivas [PRP 2006]:

1. *Física*, relacionada con el hardware y en donde las computadoras constituyen los nodos del sistema distribuido conectadas por medio de una red.
2. *Lógica*, relacionada con el software y que puede ser interpretada como un conjunto de procesos cooperantes.

En ambos casos el mecanismo de comunicación para lograr la interacción es a través del paso de mensajes. Es importante mencionar que la distribución lógica es independiente de la física ya que los procesos no necesariamente han de estar enlazados mediante una red sino que todos pueden encontrarse en una sola computadora.

Una vez identificados los elementos constitutivos de un sistema distribuido, es necesario organizarlos y establecer el rol que desempeñan. El modelo *cliente-servidor* ayuda a comprender y administrar la complejidad inherente de los sistemas distribuidos.

2.5. Modelo *cliente-servidor*

Compartir recursos es uno de los motivos principales para construir sistemas distribuidos. Por tanto, se debe distinguir entre quienes poseen tales recursos y quienes los requieren. El modelo *cliente-servidor* define dos roles que pueden ser asumidos ya sea por los procesos o por las computadoras: el rol de usuario del servicio (*cliente*) y el rol de proveedor del servicio (*servidor*).

Por lo general, se comparten recursos como impresoras, discos duros, archivos, *etc*. Tales recursos deben ser administrados mediante servidores para que puedan ser accedidos por clientes, *por ejemplo* los servidores *Web* administran un conjunto de páginas *Web* que son accedidas por clientes mediante los denominados *navegadores* (*browsers*).

Sin embargo, en muchas ocasiones es más significativo que el servidor se encargue de realizar algún tipo de procesamiento más intenso como puede ser la realización de cálculos estadísticos o científicos, el tratamiento de imágenes, la ejecución de simulaciones, *etc*.

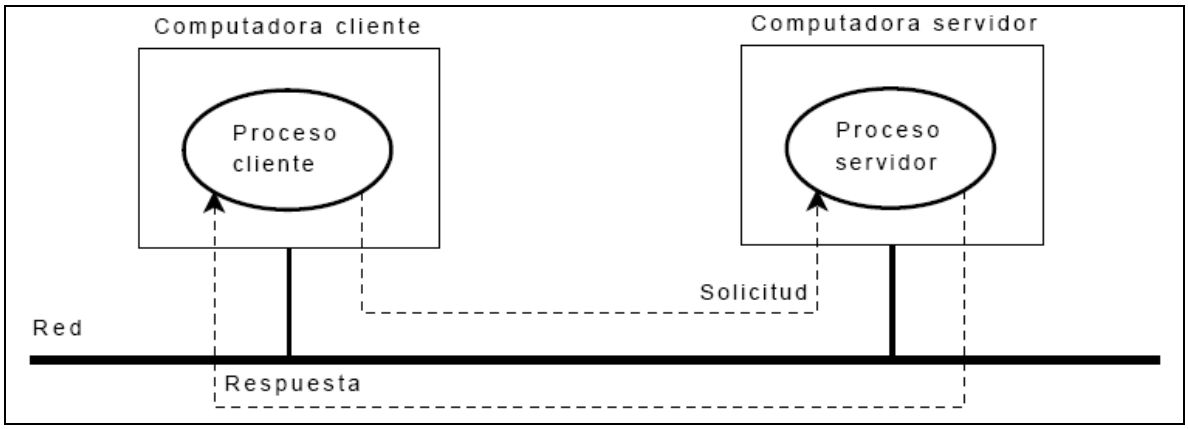

Figura 2.3. Interacción entre Cliente y Servidor [MEJ 2007]

En la figura 2.3 se representa la interacción entre cliente y servidor. Cuando un cliente requiere de un servicio, envía un mensaje de solicitud al servidor, quien espera por solicitudes entrantes y una vez recibida, realiza el trabajo de procesamiento con el fin de devolver el resultado en un mensaje de respuesta para el cliente.

Una ventaja de este modelo es que ofrece la posibilidad de proporcionar servicios de manera simultánea a múltiples clientes (concurrencia). Así, un servidor no tendría que esperar a que termine de atender un cliente para atender otro; sin embargo, aunque existen dos partes (cliente y servidor), el servicio se encuentra centralizado en un único servidor, lo cual es un cuello de botella cuando el número de clientes crece o se produce alguna falla o desconexión.

Se puede mencionar el caso del servicio *DNS*, que traduce nombres de dominio de *Internet* en direcciones de red, ¿Cómo trabajaría este servicio si estuviese implementado con una sola tabla de direcciones almacenada en una computadora específica? En estas condiciones, el único servidor debería

atender todas las solicitudes de resolución de direcciones, provocando incluso que nadie pueda usar la *Web* [T&V 2002].

La figura 2.4 ilustra una de las técnicas para enfrentar esta limitación. Consiste en que los procesos o las computadoras asuman ambos roles, tanto cliente como servidor. En el ejemplo anterior, los servidores *DNS* pueden, a su vez, ser clientes de otros servidores *DNS* cada uno administrando una base de datos de direcciones. De manera semejante, los servidores *Web* y la mayoría del resto de los servicios de *Internet* son clientes del servicio *DNS* [CDK 2001]. Así, un servicio puede estar implementado por varios servidores interactuando unos con otros.

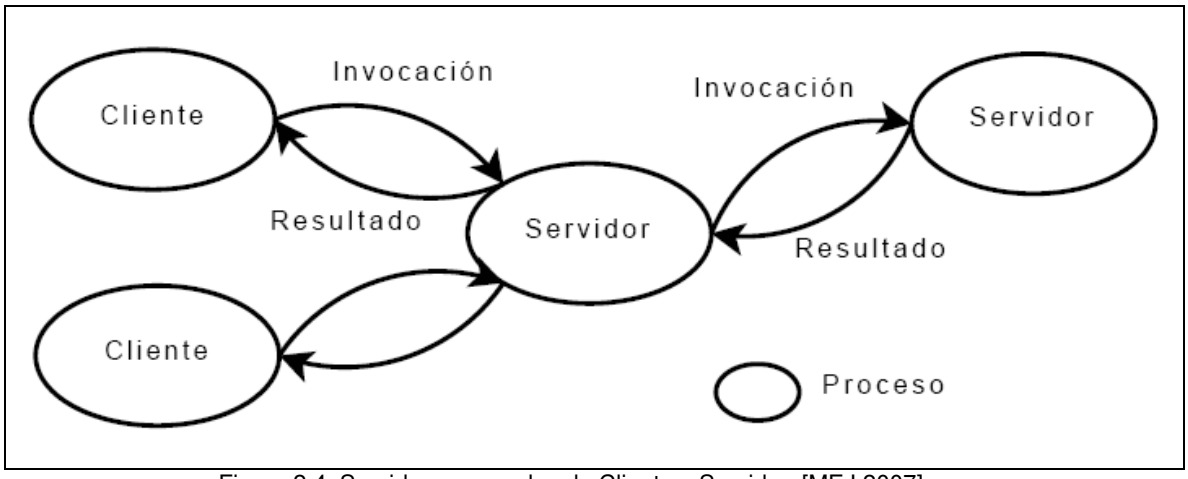

Figura 2.4. Servidores con roles de Cliente y Servidor [MEJ 2007]

2.6. Modelo de objetos distribuidos

El interés por el paradigma de *orientación a objetos* (*OO*) ha crecido rápidamente en los últimos años [JAB 1992]. Esta técnica ha probado ser conveniente para modelar la estructura y las interacciones en sistemas distribuidos cada vez más complejos. Esto es debido a las propiedades

fundamentales de objetos tales como: abstracción, encapsulamiento, modularidad, herencia y polimorfismo [G&G 1996].

En la tabla 2.2 se muestra cómo determinados conceptos del paradigma *OO* pueden ser aprovechados en sistemas distribuidos (*SD*).

Concepto	*Paradigma de OO*	*Adaptabilidad a SD*
Objeto	Entidad física o conceptual del mundo real [16]. Define un estado y comportamiento.	Los *SD* pueden estar organizados en términos de objetos, unidades de distribución que interactúan. Los objetos se refieren a procesos o computadoras, indistintamente.
Abstracción	Permite identificar y separar los atributos (estado) y métodos (comportamiento) esenciales de un objeto.	Los objetos en un *SD* también consisten de un estado (datos administrados por el objeto) así como de un comportamiento, el cual corresponde al rol que desempeña, *e.g.* cliente, servidor, o ambos.
Encapsulamiento	Agrupación de los atributos y métodos en una entidad independiente, definiendo respectivamente, un estado interno y **una interfaz que provee algunos servicios al exterior**.	Los recursos en un *SD* son administrados por servidores y pueden ser encapsulados como objetos para ser accedidos por objetos clientes.
Ocultamiento	Es una consecuencia de la encapsulación, de tal forma que el estado del objeto puede ser modificado solamente por los métodos que conforman la interfaz, la cual permite interactuar con otros objetos.	La separación entre interfaz y su implementación es clave para comunicar objetos distribuidos. Es posible colocar una interfaz en una computadora mientras que el objeto en sí reside en otra. Además, la única forma de interactuar con un objeto es la interfaz, donde se exponen solamente los detalles necesarios para el resto del sistema.
Modularidad	Debido a que cada objeto es considerado como una entidad única y aislada, se convierten en módulos naturales.	La modularidad permite desarrollar componentes de aplicación (*e.g.* clientes y servidores) que pueden ser distribuidos en varias computadoras.
Herencia	Es un mecanismo para compartir atributos y métodos con base en una relación jerárquica. Podemos ir de lo general a lo particular evitando redundancia.	La herencia en un entorno distribuido soporta la *re-utilización* de código [14]. Objetos de nuevas aplicaciones pueden aprovechar directamente las implementaciones de otros objetos en distintas ubicaciones.
Polimorfismo	Propiedad estrechamente ligada con la jerarquía de herencia. Significa que un mismo comportamiento puede tomar diferentes formas en función del objeto que se está tratando.	Ofrece capacidad de crecimiento, *i.e.* permite la *extensión* de un sistema ya que partes pueden ser añadidas para proporcionar nuevas y/o mejores funcionalidades. De manera similar, facilita la sustitución o intercambio de la implementación de un servicio (mantenimiento).
Comunicación	La comunicación (interacción) entre objetos se realiza por medio de invocaciones de métodos.	En un *SD* los procesos cooperan unos con otros a través de intercambio de mensajes. Así, las invocaciones de métodos pueden ser transformadas en mensajes.

Tabla 2.2. Conceptos del paradigma OO

La adaptabilidad del paradigma *OO* para entornos distribuidos ha determinado que la mayoría del software de sistemas distribuidos actual tienda a desarrollarse con lenguajes orientados a objetos [T&V 2002], por ejemplo aplicaciones financieras, bursátiles, colaborativas, mensajería, bases de datos compartidas, reservación de boletos aéreos, *etc*.

2.7. Middleware

Los sistemas distribuidos toman en cuenta y tratan nuevos problemas que no existen en los sistemas centralizados, entre los cuales destaca la heterogeneidad, que se manifiesta en los siguientes aspectos [PRP 2006]:

• *Aplicaciones:* los objetos que conforman una aplicación distribuida pueden estar implementados en diferentes lenguajes de programación.

• *Sistemas Operativos*: distintos sistemas operativos de las computadoras en un sistema distribuido.

• *Hardware*: diferentes arquitecturas de procesador y otros detalles técnicos que se presentan en cada computadora (por ejemplo representaciones de datos).

• *Red*: las diversas computadoras de un sistema distribuido se conectan por medio de tecnologías de red diferentes.

La heterogeneidad en cada uno de los niveles anteriores es resuelta por medio de componentes denominados *middlewares*. En la figura 2.5 se muestra la ubicación del *middleware* en un sistema distribuido:

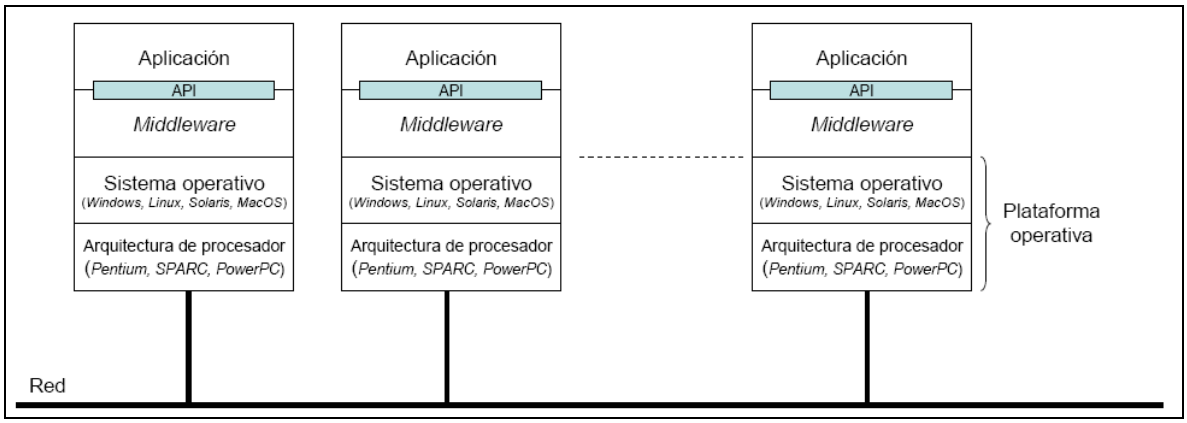

Figura 2.5. Middleware en un Sistema Distribuido [G&G 1996]

El *middleware* se ejecuta sobre diversas combinaciones de arquitectura de procesador y sistema operativo (por ejemplo *Pentium*/*Linux*, *PowerPC*/*MacOS*, *SPARC*/*Solaris*), y supera esta variedad ofreciendo igual funcionalidad en todas las computadoras por medio de una *biblioteca* de programas o *API*, útil para el desarrollo de aplicaciones distribuidas.

La implementación de un *middleware* se adhiere a un estándar con el fin de lograr *compatibilidad*. El estándar es una descripción abstracta de un comportamiento deseado y que sirve como un modelo de acuerdo con el cual diferentes productos (implementaciones) pueden ser producidos [PRP 2006]. Muchas tecnologías de *middlewares* están disponibles hoy y, en cuanto a aquellos que ofrecen soporte para el paradigma *OO*, cinco de los estándares principales para el desarrollo de sistemas distribuidos son:

- *CORBA, (Common Object Request Broker Architecture), de OMG (Object Management Group)* [OMG 2004]
- ***Java RMI, (Java Remote Method Invocation)**, de Sun Microsystems* [JRMI 1999]

• *DCOM*, (*Distributed Component Object Model*), de *Microsoft Corporation* [COMS 2008]

• *.NET Framework Remoting*, de *Microsoft Corporation* [COMS 2008]

• *Servicios Web (Servicios Web) de la Arquitectura Orientada a Servicios (SOA)* [W3C 2004]

En general, las primeras 3 tecnologías tienen una filosofía similar de funcionamiento, extienden el modelo de invocación de método en una sola computadora a invocaciones remotas entre objetos sobre distintas computadoras en un sistema distribuido. Desde el punto de vista del desarrollador, la interacción remota entre objetos es transparente, prácticamente idéntica a interacciones locales.

CORBA es resultado del esfuerzo de la *OMG* para definir un marco estándar para interoperabilidad de objetos distribuidos independientemente del lenguaje de programación.

Tanto *CORBA* como *Java RMI* son sistemas abiertos, cuyas especificaciones y documentación se encuentran libremente publicadas. Los productos de ambas especificaciones pueden ser utilizados sobre diversas plataformas de sistemas operativos, desde *mainframes* a máquinas *UNIX*, *Linux*, o *Windows* a dispositivos de mano siempre que haya una implementación para la plataforma. La especificación de *DCOM* permite que los componentes de servidor sean escritos en diversos lenguajes de programación como *C++*, *Java*, *Object Pascal* (*Delphi*), *Visual Basic* e incluso *COBOL* [GOP

2003]. *DCOM* fue muy utilizado sobre la plataforma *Windows* y es un software propietario.

En cuanto a Net Framework Remoting y los Servicios Web también son tecnologías propias de Microsoft las cuales serán después abordadas en el Estado del Arte de la presente Investigación, lo que vale resaltar es el concepto de SOA (Arquitectura Orientada a Servicios) sobre la cual se desarrollan los Servicios Web y de la cual se habla a continuación.

2.8. Arquitectura Orientada a Servicios (SOA)

La Arquitectura SOA establece un marco de diseño para la integración de aplicaciones independientes de manera que desde la red pueda accederse a sus funcionalidades, las cuales se ofrecen como servicios [MSOA 2007]. La forma más habitual de implementarla es mediante Servicios Web, una tecnología basada en estándares e independiente de la plataforma, con la que SOA puede descomponer aplicaciones monolíticas en un conjunto de servicios e implementar esta funcionalidad en forma modular.

¿Qué es un servicio exactamente? Un servicio es una funcionalidad concreta que puede ser descubierta en la red y que describe tanto lo que puede hacer como el modo de interactuar con ella.

Desde la perspectiva de la empresa, un servicio realiza una tarea concreta: puede corresponder a un proceso de negocio tan sencillo como introducir o extraer un dato como “Código del Cliente”. Pero también los

servicios pueden acoplarse dentro de una aplicación completa que proporcione servicios de alto nivel, con un grado de complejidad muy superior –por ejemplo, “introducir datos de un pedido”-, un proceso que, desde que comienza hasta que termina, puede involucrar varias aplicaciones de negocio.

La estrategia de orientación a servicios permite la creación de servicios y aplicaciones compuestas que pueden existir con independencia de las tecnologías subyacentes. En lugar de exigir que todos los datos y lógica de negocio residan en un mismo ordenador, el modelo de servicios facilita el acceso y consumo de los recursos de IT a través de la red. Puesto que los servicios están diseñados para ser independientes, autónomos y para interconectarse adecuadamente, pueden combinarse y recombinarse con suma facilidad en aplicaciones complejas que respondan a las necesidades de cada momento en el seno de una organización. Las aplicaciones compuestas (también llamadas “dinámicas”) son lo que permite a las empresas mejorar y automatizar sus procesos manuales, disponer de una visión consistente de sus clientes y socios comerciales y orquestar sus procesos de negocio para que cumplan con las regulaciones legales y políticas internas. El resultado final es que las organizaciones que adoptan la orientación a servicios pueden crear y reutilizar servicios y aplicaciones y adaptarlos ante los cambios evolutivos que se producen dentro y fuera de ellas, y con ello adquirir la agilidad necesaria para ganar ventaja competitiva.

2.8.1. SaaS

Otro concepto muy ligado a SOA es la noción de “Software como Servicio” (Saas, “Software as a Service”). [MSOA 2007] En pocas palabras, SasS puede definirse como “software que se pone en explotación en la modalidad de servicio gestionado y que al cual se accede a través de Internet”.

El concepto de SaaS suele asociarse con los proveedores de servicios de aplicación de los años 90, que ofrecían aplicaciones “empaquetadas” a los usuarios corporativos a través de Internet.

Estos primeros intentos de poner en marcha soluciones de Software a través de Internet tenían más en común con las aplicaciones corporativas tradicionales (las que se instalan y utilizan dentro de la red interna de las empresas) que con las actuales aplicaciones SaaS en muchos aspectos, tales como el modelo de licencia y la arquitectura. Puesto que esas aplicaciones se crearon en principio como aplicaciones para un solo destinatario, su capacidad para compartir datos y procesos con otras aplicaciones estaba muy limitada y tendían a ser escasamente atractivas en comparación con sus equivalentes de instalación en local.

Hoy día las aplicaciones SaaS pretenden aprovechar las ventajas de la centralización a partir de una arquitectura de instancia única con múltiples usuarios y ofrecer una experiencia con funcionalidades avanzadas que compitan con ventaja frente a las aplicaciones instaladas localmente. Una aplicación SaaS normalmente la ofrece un proveedor de forma directa o un

intermediario (llamado “agregador”) que empaqueta ofertas SaaS de distintos proveedores y las ofrece como una plataforma unificada de e aplicaciones o una suite de servicios de aplicación.

A diferencia del modelo de licencias habitual del software que se instala en las empresas, el acceso a las aplicaciones SaaS se suele basar en un modelo de suscripción, donde los clientes pagan una tarifa por adelantado para utilizarlas. Las estructuras de precios varían de unas aplicaciones a otras: algunos proveedores aplican una tarifa plana con acceso ilimitado a diversas funcionalidades de las aplicaciones, y otros aplican tramos tarifarios que dependen del nivel de utilización.

SaaS además se posiciona como uno de los pilares del desarrollo de la orientación a servicios. A los efectos de este documento, nos vamos a referir de forma genera a SOA, incluyendo en este concepto tanto los servicios implantados en local como los alojados en Internet. Consideramos que SaaS es un componente fundamental en cualquier estrategia SOA de un cliente.

2.8.2. BPM

El concepto de BPM (Business Process Management) está también muy ligado a SOA. BPM es una disciplina de gestión que combina una visión centrada en procesos y de integración de funcionalidades que pretende mejorar la efectividad de las organizaciones [MSOA 2007]. Una solución BPM dispone de los medios necesarios para la realización efectiva de estos procesos así como las funcionalidades necesarias para que los gestores de las empresas

puedan controlar y modificar los flujos de trabajo (“workflows”) tanto manuales como automáticos.

La gestión de procesos de negocio tiene sus orígenes en los Sistemas de Gestión de Calidad Total y la reingeniería de procesos. Puesto que les añade un marco tecnológico de desarrollo, BPM es más que una combinación de estas disciplinas: BPM es una disciplina de gestión de procesos dirigida mediante Tecnologías de Información, capaz de mejorar la agilidad organizativa y que mejora la capacidad de las personas para introducir cambios en los procesos e innovar de forma rápida. Por consiguiente, BPM permite el alineamiento de las tecnologías de información con las actividades de negocio, tanto en el seno de la propia organización como fuera de ella, con socios comerciales, proveedores y clientes.

Los procesos de negocio pueden ser estructurados o no estructurados, dependiendo de hasta qué punto los pasos que comprenden son pasos bien establecidos – y susceptibles, por tanto, de automatización- o intercambiables, y generalmente ejecutados por personas solamente o por personas que interactúan con sistemas. Las personas son una parte esencial de prácticamente cualquier proceso de negocio: aplican las soluciones y disponen de la visión que hace avanzar a una empresa, por lo que el objetivo debe ser aumentar su capacidad para crear e innovar y ser más productivas (y no “hacer reingeniería” pretendiendo colocar a las personas fuera de los procesos).

Aunque BPM puede considerarse como una entidad al margen de las iniciativas SOA, la capacidad para definir nuevos procesos de negocio de forma flexible y rápida es mucho mayor si los recursos de los sistemas de IT se exponen en la forma de orientación a servicios.

2.8.3. Beneficios de SOA

Los beneficios de SOA para una organización se plasman a dos niveles distintos: al del usuario corporativo y a nivel de la organización de IT.

Desde el punto de vista de la empresa, SOA permite el desarrollo de una nueva generación de aplicaciones dinámicas que resuelven una gran cantidad de problemas de alto nivel, fundamentales para el crecimiento y la competitividad. Las soluciones SOA permiten entre otras cosas:

• **Mejorar la toma de decisiones.** Al integrar el acceso a los servicios e información de negocio dentro de un conjunto de aplicaciones dinámicas compuestas, los directivos disponen de más información y de mejor calidad (más exacta y actualizada). Las personas, procesos y sistemas que abarcan múltiples departamentos pueden introducirse de forma más directa en una panorámica unificada, lo que permite conocer mejor los balances de costos y beneficios que se producen en las operaciones de negocio que se realizan a diario. Y al disponer de mejor información en un tiempo menor, las organizaciones pueden reaccionar de manera más ágil y rápida cuando surgen problemas o cambios.

• **Mejorar la productividad de los empleados.** Un acceso óptimo a los sistemas y la información y la posibilidad de mejorar los procesos permiten a las empresas aumentar la productividad individual de los empleados. Estos pueden dedicar sus energías a los procesos importantes, los que generan valor añadido y a actividades de colaboración, semiestructuradas, en vez de aceptar las limitaciones y restricciones impuestas por los sistemas de IT rígidos y monolíticos. Más aún: puesto que los usuarios pueden acceder a la información en los formatos y modalidades de presentación (web, cliente avanzado, dispositivo móvil), que necesitan, su productividad se multiplica en una gran cantidad de escenarios de uso, habituales o nuevos.

• **Potenciar las relaciones con clientes y proveedores**. Las ventajas de SOA trascienden las fronteras de la organización. Los beneficios que ofrece SOA trascienden los límites de la propia organización. Los procesos de fusión y compra de empresas se hacen más rentables al ser más sencilla la integración de sistemas y aplicaciones diferentes. La integración con partners comerciales y la optimización de los procesos de la cadena de suministro son, bajo esta perspectiva, objetivos perfectamente asequibles. Con SOA se puede conseguir mejorar la capacidad de respuesta a los clientes, habilitando por ejemplo portales unificados de servicios. Si los clientes y proveedores externos pueden disponer de acceso a aplicaciones y servicios de negocio dinámicos, no solamente se permite una colaboración avanzada, sino que se aumenta la satisfacción de clientes y proveedores. SOA permite flexibilizar los procesos críticos de compras y gestión de pedidos –habilitando modalidades como la subcontratación de ciertas actividades internas- superando las restricciones

impuestas por las arquitecturas de IT subyacentes, y con ello consiguiendo un mejor alineamiento de los procesos con la estrategia corporativa.

SOA contribuye también a documentar el modelo de negocio de la empresa y a utilizar el modelo de negocio documentado para integrar en él y dar respuesta a las dinámicas de cambio que se produzcan y optimizarlo de acuerdo con ellas.

Desde el punto de vista de los departamentos de IT, la orientación a servicios supone un marco conceptual mediante el cual se puede simplificar la creación y mantenimiento de sistemas y aplicaciones integradas, y una fórmula para alinear los recursos de IT con el modelo de negocio y las necesidades y dinámicas de cambio que le afectan.

• **Aplicaciones más productivas y flexibles.** La estrategia de orientación a servicios permite a IT conseguir una mayor productividad de los recursos de IT existentes –como pueden ser las aplicaciones y sistemas ya instalados e incluso los más antiguos- y obtener mayor valor de ellos de cara a la organización sin necesidad de aplicar soluciones de integración desarrolladas ex profeso para este fin. La orientación a servicios permite además el desarrollo de una nueva generación de aplicaciones compuestas que ofrecen capacidades avanzadas y multifuncionales para la organización con independencia de las plataformas y lenguajes de programación que soportan los procesos de base. Más aún: puesto que los servicios son entidades

independientes de la infraestructura subyacente, una de sus características más importantes es su flexibilidad a la hora del diseño de cualquier solución.

• **Desarrollo de aplicaciones más rápido y de menor costo**. El diseño de servicios basado en estándares facilita la creación de un repositorio de servicios reutilizables que se pueden combinar en servicios de mayor nivel y aplicaciones compuestas en respuesta a nuevas necesidades de la empresa. Con ello se reduce el costo del desarrollo de soluciones y de los ciclos de prueba, se eliminan redundancias y se consigue su puesta en valor en menos tiempo. Y el uso de un entorno y un modelo de desarrollo unificados simplifica y homogeniza la creación de aplicaciones, desde su diseño y prueba hasta su puesta en marcha y mantenimiento.

• **Aplicaciones más seguras y manejables**. Las soluciones orientadas a servicios proporcionan una infraestructura común (y una documentación común también) para desarrollar servicios seguros, predecibles y gestionables. Conforme van evolucionando las necesidades de negocio, SOA facilita la posibilidad de añadir nuevos servicios y funcionalidades para gestionar los procesos de negocio críticos. Se accede a los servicios y no a las aplicaciones, y gracias a ello la arquitectura orientada a servicios optimiza las inversiones realizadas en IT potenciando la capacidad de introducir nuevas capacidades y mejoras. Y además, puesto que se utilizan mecanismos de autenticación y autorización robustos en todos los servicios –y puesto que los servicios existen de forma independiente unos de otros y no se interfieren entre ellos- la estrategia de SOA permite dotarse de un nivel de seguridad superior.

2.8.4. Como se resuelven los retos de SOA

Embarcarse en un proyecto de SOA supone tener que resolver una serie de retos, tanto a nivel organizativo como técnico, y estos retos pueden convertirse en verdaderas barreras insuperables si se ha partido de la idea de que SOA es el remedio para toda clase de males.

Para que las iniciativas de adopción de SOA tengan un fin satisfactorio, hay que asegurarse de que se cumplen una serie de condiciones indispensables:

• **Definir claramente los objetivos de negocio**. El primer paso a la hora de adoptar SOA es identificar con claridad los problemas o retos empresariales más prioritarios. Cuando más precisa sea esa formulación, más fácilmente se podrá delimitar la dirección y el alcance de cualquier proyecto SOA. Disponer de una visión y un rumbo claro desde el principio hará mucho más fácil la ejecución de procesos cuya esencia es la integración de múltiples funciones.

• **Definir claramente el alcance del proyecto SOA**. El objetivo de cualquier proyecto SOA no debe consistir en renovar de forma indiscriminada y masiva toda la infraestructura de IT. Este tipo de megaproyectos fracasan a la hora de implementarlos porque cuando por fin se ha conseguido crear la solución, las condiciones del negocio suelen haber cambiado tanto que los problemas que ahora deben resolverse ya no tienen mucho que ver con aquellos que se pretendían resolver cuando se inició el proyecto. El objetivo real de cada

iniciativa SOA debe ser responder a necesidades concretas de negocio y crear soluciones en pasos discretos, incrementales e iterativos.

• **Evitar introducir SOA sin motives reales que lo justifiquen**. La adopción de SOA no debe considerarse una necesidad tecnológica, sino organizativa: debe responder a las necesidades de la organización. Si la introducción de SOA solamente responde al puro gusto por disponer de SOA y se empiezan a crear servicios sin un significado de negocio claro, sin la granularidad adecuada o con demasiadas interconexiones, el resultado será una implementación excesivamente compleja, inmanejable y tremendamente costosa.

• **Gestionar el proceso**. Los servicios y aplicaciones se corresponden con procesos y los outputs de información deseados a través de las diversas áreas funcionales de la organización. Puesto que representan procesos compartidos, es necesario que se les asigne un propietario para que puedan inventariarse y gestionarse a fin de garantizar que cumplen en todo momento con las directivas corporativas y responden adecuadamente a las necesidades que los justifican.

2.8.5. Elementos esenciales de una Arquitectura Orientada a Servicios

En las Arquitecturas Orientadas a Servicios, el elemento básico es el servicio. Pero únicamente con este concepto, no podríamos diseñar una arquitectura SOA. Cuatro son los elementos esenciales necesarios para la construcción de una Arquitectura Orientada a Servicios:

1. **Operación:** Es la unidad de trabajo o procesamiento en una arquitectura SOA.
2. **Servicio:** Es un contenedor de lógica. Estará compuesto por un conjunto de operaciones, las cuales las ofrecerá a sus usuarios.
3. **Mensaje:** Para poder ejecutar una determinada operación, es necesario un conjunto de datos de entrada. A su vez, una vez ejecutada la operación, esta devolverá un resultado. Los mensajes son los encargados de encapsular esos datos de entrada y de salida.
4. **Proceso de negocio:** Son un conjunto de operaciones ejecutadas en una determinada secuencia (intercambiando mensajes entre ellas) con el objetivo de realizar una determinada tarea.

Figura 2.6. Elementos de una Arquitectura SOA [MOL 2004]

Por lo tanto, una aplicación SOA estará formada por un conjunto de procesos de negocio. A su vez esos procesos de negocio estarán compuestos por aquellos que servicios que proporcionan las operaciones que se necesitan ejecutar para que el proceso de negocio llegue a buen término. Por último para

ejecutar esas operaciones es necesario el envío de los datos necesarios mediante los correspondientes mensajes.

CAPITULO III

ESTADO DEL ARTE

Teniendo los conocimientos previos alrededor del problema dados en el marco teórico, se realizará un estudio profundo y más amplio sobre el tema expuesto en la presente investigación, mediante una revisión de las arquitecturas distribuidas existentes más importantes relacionadas al presente problema de estudio.

3.1. Taxonomía del problema

Siendo el problema que se aborda en la investigación el de crear una Solución Distribuida orientada a Servicios para la transferencia de datos entre agentes del Sistema de EPS, utilizando para esto los Servicios Web; se le puede clasificar dentro del grupo de las aplicaciones distribuidas (ver página 19) debido a que se van a manejar procesos que no se encuentran centralizados, sino que se comparten entre las EPS y la SEPS.

Ya que localizamos la naturaleza de la arquitectura del problema, corresponde ver que tipo de tecnología distribuida abarca. Como la solución se plantea realizarla con Servicios Web, entonces estamos incursionando en el área de la Arquitectura Orientada a Servicios (ver pagina 34), dejando de lado

la opción de componentes (se explicará más adelante en que consiste esta tecnología).

En la figura 3.1 puede apreciarse las clases de Arquitecturas para la construcción de software de la cual nuestro problema engloba tanto objetos distribuidos como Servicios Web.

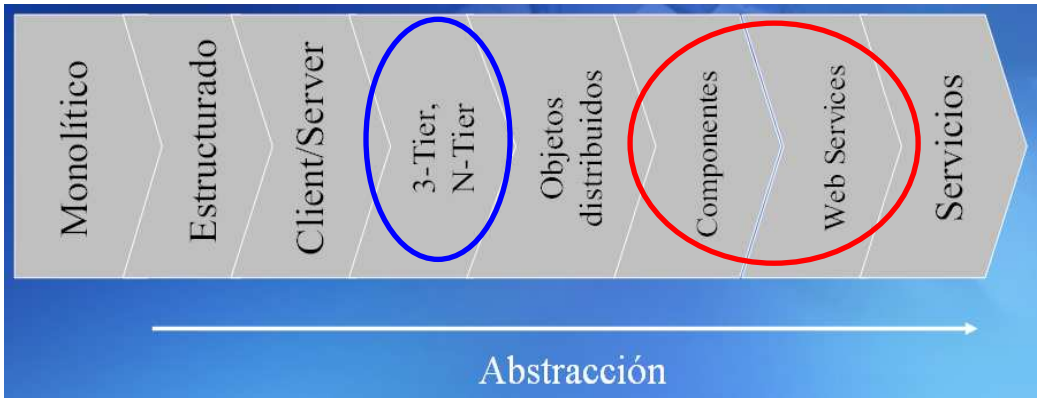

Figura 3.1. Tipos de Arquitecturas de Sistemas de Información

3.2. Tecnologías involucradas en la solución del problema

Se verán las tecnologías distribuidas que pueden aplicarse para la resolución del problema para posteriormente realizar un análisis comparativo entre éstas y concluir que son los Servicios Web la solución adecuada con el problema expuesto en la investigación.

3.2.1. Introducción a las Tecnologías de Objetos Distribuidos

Para optar por una arquitectura distribuida como solución del problema, se deben considerar distintos aspectos [MSDAS 2006], entre otros:

- Escalabilidad
- Performance
- Interoperabilidad

- Seguridad
- Bajo Costo
- Rápida Implementación

A continuación mencionamos las principales tecnologías distribuidas existentes y que cumplen al menos con algunos de los aspectos mencionados arriba y de las que seleccionaremos la más adecuada para dar solución al problema:

3.2.2. CORBA

CORBA son las siglas de Common Object Request Broker Architecture.

En el año 1989, se formó la OMG (Object Management Group) para propiciar los sistemas de objetos distribuidos que garanticen la interoperabilidad. Para ello, trabajó dentro de la filosofía de los sistemas abiertos.

El OMG fue quien inició la terminología ORB. Y en el año 1990, un grupo de empresas acuerdan una especificación para una arquitectura de este tipo, que llamaron CORBA.

En 1996, se definió la especificación CORBA 2.0. A través de ella se dictan los estándares para implementar la comunicación entre objetos definidos en lenguajes diferentes. Estos estándares reciben el nombre de GIOP (General Inter-ORB Protocol) y la idea era que este GIOP puede ejecutarse sobre cualquier capa de transporte.

La especialización TCP/IP de GIOP es IIOP (Internet Inter-ORB protocol). En otras palabras: GIOP es el protocolo base, y IIOP lo “adapta” a TCP/IP. Los ORB del cliente y del servidor interactúan a través de mensajes GIOP cuando quieren ubicar un objeto. El ORB del cliente local, una vez ubicado el objeto remoto, le pasará al cliente una referencia para que éste pueda invocar los métodos del objeto remoto.

El paquete GIOP, tiene un header donde se almacena la medida del mensaje, la versión de GIOP, el ordenamiento de los bytes. Los datos se alinean según la implementación usada: no hay especificación global. Los datos son almacenados en formato CDR, lo cual, obviamente, debe conocer el receptor para deserializarlo adecuadamente.

CORBA utiliza un formato para las referencias a objetos remotos, el IOR (Interoperable Object Reference) que se puede usar si el objeto es activable o no.

Nombre de tipo de interfase IDL	Protocolo y dirección			Clave de Objeto	
Identificador de repositorio de interfase	IIOP	Nombre del dominio del host	Número de puerto	Nombre de adaptador	Nombre de objeto

Tabla 3.1. Interoperable Object Referente [MOL 2004]

En el primer campo del IOR, se especifica el nombre de tipo de interfaz IDL del objeto CORBA. Si el ORB tiene un repositorio de interfaz, este nombre debe coincidir con el identificador en el repositorio.

Acción de un cliente que requiere un objeto remoto usando CORBA. El cliente se linkea en tiempo de compilación, a un stub que al activarse actúa como un proxy.

Busca en el repositorio dinámicamente para invocar el objeto remoto. Cuando el cliente quiere invocar el objeto remoto, llama al método que hace el binding, lo que provoca que el runtime CORBA determine donde está el objeto y fuerza al servidor para que lo active. Cuando desde el objeto llega la notificación sobre su inicialización, el servidor retorna al cliente, una referencia al objeto. Y entonces si: el cliente puede invocar el método remoto.

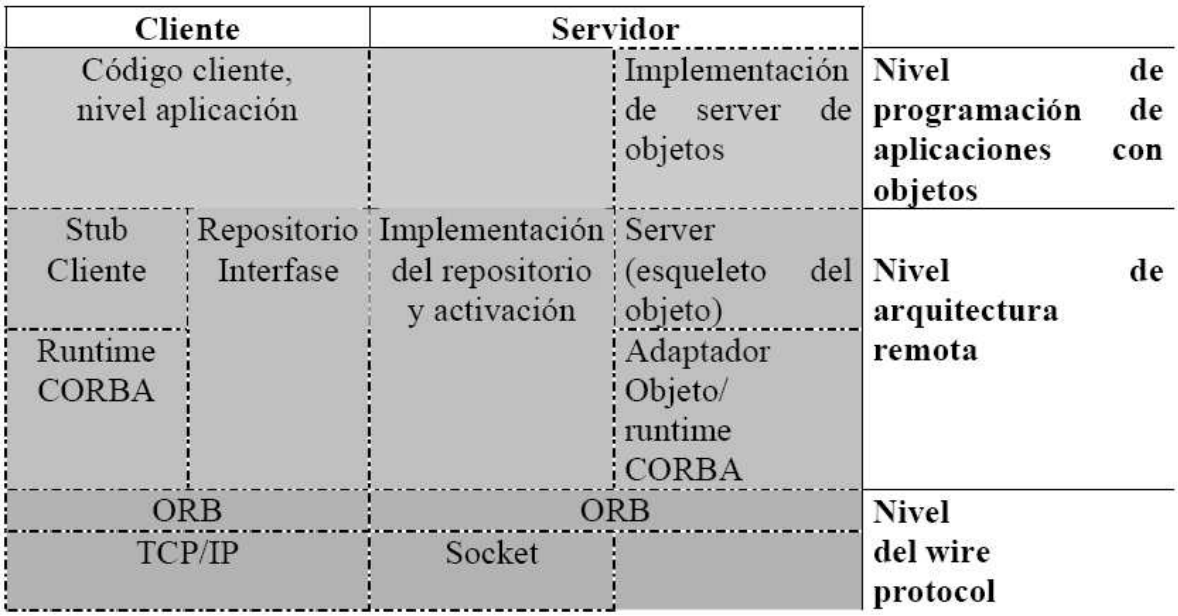

Figura 3.2. Arquitectura CORBA [MOL 2004]

3.2.3. COM, COM+ y DCOM

Microsoft Component Services se compone de las siguientes tecnologías:

- **COM** (Component Object Model)
- **DCOM** (Distributed Component Object Model)

- **MTS** (Microsoft Transaction Server)
- **MIIS** (Microsoft Internet Information Server)
- **MSMQ** (Microsoft Message Queue)

A través de estas tecnologías, Microsoft trata con la construcción e implementación de aplicaciones distribuidas. Cuando en 1993 surge COM, tecnología de Microsoft, se avanzó en la posibilidad de comunicar aplicaciones escritas en distintos lenguajes.

En COM, el cliente llama a funciones en el server a “nivel binario”, y por lo tanto, no necesita conocer en qué código fue implementado. También a través de COM se podía acceder a cierta funcionalidad del sistema operativo, como manejo de transacciones y queueing.

Pero tiene dos desventajas:

- Requiere participar de la infraestructura de la aplicación. Se debe analizar el marshalling que usa y adaptarlo si es necesario. Las diferencias las debe tener en cuenta el desarrollador para subsanarlas.
- COM trata con las aplicaciones a través de interfases externas, es decir, que no comparte la implementación interna. Por ejemplo si las aplicaciones a comunicarse tratan el tipo string de distinta forma, lo debe prever el desarrollador para zanjar las diferencias.

Para trabajar con estas diferencias, el aporte de las características de la programación orientada a objetos es innegable. De la aplicación local en COM, se creció al ambiente distribuido con DCOM. DCOM presenta una arquitectura compleja.

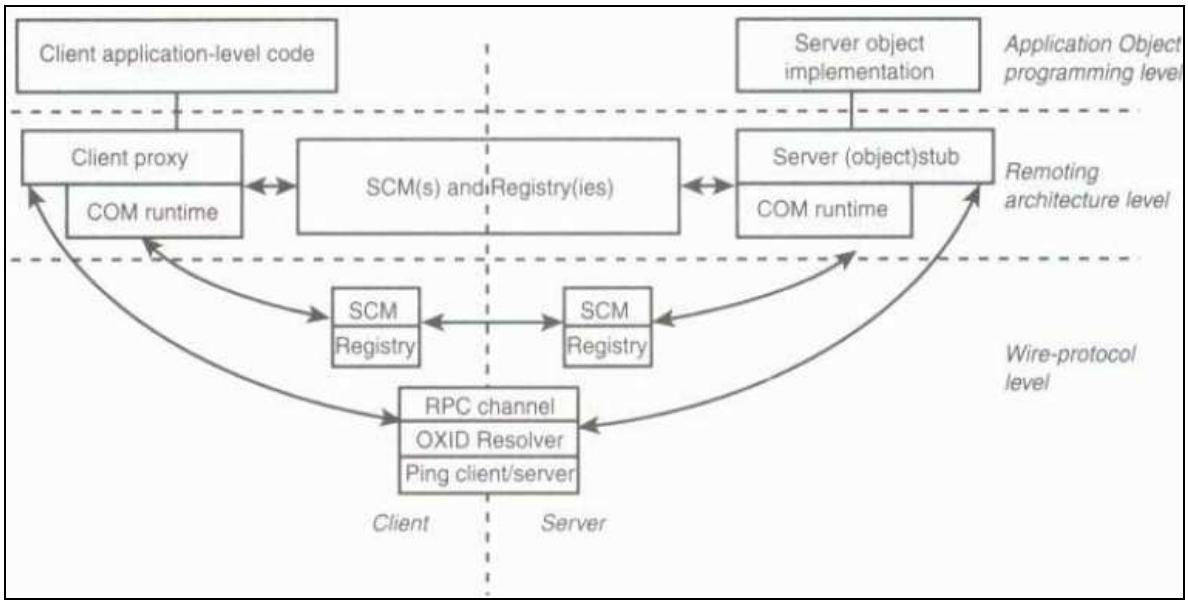

Figura 3.3. Arquitectura COM y DCOM [SCR002]

En DCOM cuando el cliente necesita acceder a un objeto remoto, se crea un proxy local. A través del SCM (Service Control Manager) local se tiene la información del sistema remoto. También puede ofrecer esta información la Registry. El SCM local contacta al SCM remoto. Este activa el objeto remoto y pasa la referencia del objeto remoto al sistema local.

Esa referencia llega al proxy del objeto y el cliente puede crear instancias del objeto remoto. En este punto, el cliente está en condiciones de tratar con el objeto remoto usando el proxy y RPC.

DCOM usa diferentes tipos de mensajes para invocar métodos remotos. El paquete OBJREF se usa para codificar la respuesta de un sistema remoto, para identificar la instancia particular de un objeto. El paquete ORPCTHIS es usado por el cliente para invocar un método.

En cuanto a escalabilidad, DCOM presenta algunos problemas, pues su mecanismo de garbage collection genera mucho tráfico. Para garantizar que todos los participantes de una invocación remota están activos mientras se resuelve la llamada al método, deben enviar “ping” al server cada dos minutos.

Cuando el sistema sospecha que se perdió una conexión con el cliente, por ejemplo ante la pérdida de un ping, espera dos períodos de time out. Si luego de ese tiempo el cliente no respondió con un ping, el server destruye el objeto remoto y restituye los recursos.

DCOM, además, intenta implementar transparencia en cuanto a la ubicación del objeto. Además considera que los clientes no necesitan saber si el objeto está activado, local o remoto. Estas características requieren un constante manejo de estados lo que hace muy compleja la arquitectura.

En cuanto a la seguridad, DCOM permite seleccionar el grado de seguridad en la comunicación. Se puede transmitir texto claro, encriptar el header y los datos de la llamada al método y otras alternativas.

MTS soporta escalabilidad, seguridad, manejo de transacciones, en fin: todo lo necesario para que una aplicación en un servidor pueda crecer sobre una capa de servicios en forma estándar. Es el primer software comercial que combinó transacciones con componentes, en 1996.

Cada aplicación basada en MTS se construye desde componentes COM y los clientes acceden remotamente a través de DCOM. Mientras DCOM provee una comunicación asincrónica usando RPC, las aplicaciones basadas en Web usan, preferentemente, HTTP.

MSMQ provee dentro de Windows NT server la forma de implementar comunicación asincrónica y no bloqueante. Así, una aplicación puede enviar mensajes a otra aplicación sin esperar la respuesta. Los mensajes se envían a una cola.

MSMQ basa su acceso en COM, se integra con MTS, soporta múltiples plataformas, maneja prioridades para los mensajes, puede firmar digitalmente y encriptar mensajes, etc. Toda esta tecnología puede trabajar junta. Supongamos que queremos implementar un sistema usando Microsoft Component Services. Puede haber uno o más componentes COM que tienen la lógica de negocio.

Acceden a bases de datos, por ejemplo, para realizar consultas. A su vez estas componentes COM pueden trabajar bajo MTS para dar soporte transaccional entre bases de datos y otros servicios.

Puede usar MSMQ si la componente quiere enviar un mensaje sin esperar la respuesta. Puede haber estaciones Windows, clientes que se comunican a través de DCOM.

COM+ provee soporte automático para desarrollar objetos que participan dentro de transacciones. El desarrollador “marca” el objeto como que va a requerir una transacción.

Cuando el objeto se activa, crea una. Supongamos que el objeto quiere hacer cambios a una base de datos. Si todos los objetos que intervienen están de acuerdo, la transacción puede llegar a buen término, informan a COM+ que hace un commit de la transacción. Si no están de acuerdo, COM+ aborta la transacción.

Mientras DCOM surge desde Microsoft, CORBA es creado por una coalición de empresas, para proveer un estándar abierto para ambientes heterogéneos. CORBA permite independizarse de la plataforma y ofrece ubicuidad. Existe desde 1990 y la implementación “comercial” data de 1992. DCOM estaba disponible en beta en 1992.

CORBA tiene las ventajas inherentes a la tecnología Java. Hay muchos ORB’s Java lo que hace posible escribir clientes y servidores Java que se pueden ejecutar en cualquier plataforma donde la máquina virtual Java esté disponible.

DCOM especifica la interfase entre objetos, separando interfases de implementación y buscando API's para encontrar dinámicamente las interfases, cómo cargar e invocar. DCOM tiene una API compleja. Además, DCOM no soporta herencia múltiple (CORBA si), si no múltiples interfases. Está basado en componentes ActiveX y trabaja en forma nativa con TCP/IP, Java y HTTP.

DCOM ofrece soporte para transacciones distribuidas, messaging, seguridad, multithreading. CORBA ofrece servicios como naming, eventos, life cycle, transacciones distribuidas, seguridad, persistencia.

3.2.4. RPC

RPC es una comunicación entre máquinas que están interconectadas. Si se usa TCP/IP, esta conexión se implementará a través de direcciones IP y números de puertos.

El cliente debe determinar dirección IP o hostname del servidor con la que se desea conectar, y el puerto que lo relaciona con la aplicación que le interesa. El servidor debe tener registrada la relación entre puertos y aplicaciones, para que, cuando se recibe una petición, pueda ejecutarse el procedimiento apropiado.

El endpoint mapper es una implementación específica de DCOM que hace esta función, manteniendo una tabla que relaciona puertos e interfases a

aplicaciones específicas. En CORBA esta relación la establece el ORB (Object Request Broker).

SUN RPC fue diseñado inicialmente para la comunicación Cliente Servidor para NFS sobre SUN (ONC RPC). Usa un lenguaje de interfaz que es XDR y un compilador de interfaz que es rpcgen.

Este paradigma se basa en que tenemos dos computadoras unidas por algún vínculo de comunicación donde una de ellas quiere ejecutar un procedimiento que esta en la otra. Normalmente usa IPC (interprocess communication) para implementarse. Es una comunicación basada en mensajes. Se puede usar sobre TCP o UDP.

En el sistema remoto hay un proceso demonio RPC que esta constantemente escuchando en el port para ver si llega algo. Este mensaje que llega tiene una estructura bien definida: el identificador de la función que quiere ejecutar y los parámetros que quiere pasar a esa función.

RPC permite que los programas llamen a los procedimientos ubicados en otra máquina, imitando la función de llamada a un procedimiento local: el programa invoca los procedimientos indicando los parámetros, espera que se ejecute y al terminar retorna el control a la dirección siguiente a la llamada.

En el caso de los sistemas distribuidos, entre los parámetros se debe indicar la referencia a otra maquina y lo que complica el esquema es que las

maquinas que contienen el programa y el procedimiento, pueden no ser iguales: pueden tener distinta arquitectura, son distintos espacios de direcciones, etc.

La máquina que contiene el servidor debe recibir una solicitud que debe poder comprender, de la misma manera que el cliente debe poder comprenderlos mensajes devueltos por el servidor. Otro punto es como reacciona el sistema ante fallas, tanto de la máquina que contiene el proceso servidor como la del cliente.

Por cada servicio se almacena en el servidor, el número de programa, la versión y el número de puerto local por el cual se atenderá la solicitud de servicio. Al arrancar el servidor, le da estos datos a su demonio portmapper. De esta manera el cliente antes de mandar una petición, solicita al portmapper cuál es el puerto indicado, número de programa y versión.

Los parámetros pueden ser pasados por valor, por referencia o por copia/restauración. Los dos primeros ya son conocidos por Uds. La copia/restauración permite que quien recibe la llamada copia la variable en la pila (como si fuera por valor) y cuando retorna sobrescribe el valor original.

El objetivo es la transparencia: que el programa no sepa que se está ejecutando en otra máquina (que sea como un llamado a procedimiento local).

Para lograrlo veamos como se procede. Sea “A” el programa que corre en una máquina y supongamos que el procedimiento a ejecutar es read, que tiene como parámetros el descriptor de archivos, un puntero a la estructura donde se deben transferir los datos y un contador de bytes. Supongamos que queremos leer un archivo que está en otra máquina (un servidor de archivos), o sea que esta invocación será un llamado a procedimiento remoto.

En la máquina donde esta ejecutándose A hay una versión de read que se llama resguardo de cliente. Esta versión toma los parámetros, los mete en un mensaje y le pide al kernel que lo envíe al servidor. Luego de invocar send, esta versión invoca a receive y se bloquea hasta que regrese la respuesta.

La función del resguardo de cliente es semejante a la de un proxy. Si bien se comporta como un procedimiento local del cliente, empaqueta el identificador del remoto y los argumentos, y se envía como una petición al servidor. Al retornar el mensaje de respuesta desempaqueta los resultados.

Cuando el mensaje llega al servidor es enviado a un resguardo del servidor que está conectado con el servidor real. Este resguardo, que estaba esperando la llegada del mensaje, habrá ejecutado un receive y estaba bloqueado, a la espera. Invoca entonces al servidor a la manera usual, siendo una invocación local.

Cuando el servidor termina retorna al resguardo del servidor quien empaqueta los resultados en un mensaje y llama al send para remitirlo al cliente. Vuelve a su ciclo llamando a receive para esperar nuevos mensajes.

Al regresar a la máquina del cliente, el kernel ve que está dirigido al cliente (en realidad, al resguardo del cliente pero el kernel no lo sabe). El resguardo de cliente desempaqueta el resultado, la copia al verdadero proceso cliente y regresa de modo usual.

El proceso cliente cuenta con los datos y no se enteró que el llamado se hizo en modo remoto. Esta transparencia es la que se quiere lograr: el cliente no usó send y receive. Estos son procedimientos de biblioteca tratados por el resguardo de cliente.

Cliente	Llamada a procedimiento de resguardo
Resguardo del cliente	Prepara el buffer de mensajes Ordena parámetros dentro del buffer Llena los campos de encabezado del mensaje Señala al núcleo
Núcleo	Cambio de contexto al núcleo Copia el mensaje al núcleo Determina la dirección de destino Coloca la dirección en el encabezado del mensaje Establece la interfaz de la red Inicia el cronómetro
Servidor	realiza el servicio
Resguardo del servidor	Llama al servidor Establece los parámetros en la pila Desempaqueta los parámetros
Núcleo	Cambio de contexto del resguardo del servidor Copia el mensaje al resguardo del servidor Ve si el resguardo está esperando Decide a cuál resguardo dárselo Verifica la validez del paquete Interrupción del proceso

Tabla 3.2. Actividades en el servidor RPC

La tabla 3.2 se debe leer de arriba hacia abajo. Por ejemplo: programa cliente invoca al resguardo de cliente. Luego se realizan las actividades asociadas al resguardo de cliente (Prepara el buffer, ordena parámetros, etc)

La tabla de actividades en el servidor debe leerse de abajo hacia arriba. Por ejemplo: al llegar la petición desde el cliente, en el servidor hay una interrupción al proceso que se estaba ejecutando, se verifica la validez del paquete, etc.). Cuando se realiza el servicio, se reinvierte el camino: se llama al resguardo del servidor, se toman los parámetros de la pila, se empaquetan los resultados, etc.

El distribuidor selecciona uno de los procedimientos de resguardo, donde se desempaquetan argumentos, se llama al procedimiento de servicio correspondiente y se empaquetan los datos. Estos procedimientos de servicio implementan lo que se define en la interfaz.

El resguardo de cliente debe empaquetar los parámetros para enviarlos al servidor. El ordenamiento de parámetros, como se llama al empaquetamiento de parámetros, puede ser muy simple si tratamos con máquinas del mismo tipo. Pero en un sistema distribuido puede haber máquinas diferentes que se quieren comunicar diferentes códigos de caracteres (EBCDIC o ASCII), diferente representación de enteros (complemento a 2), etc.

Para solucionarlo se debe conocer la organización del mensaje, la identidad del cliente y cada máquina debe contar con todas las conversiones de los formatos de los demás.

Entre diferentes alternativas, podemos analizar una donde el cliente usa su formato, original. Cuando llega el mensaje al resguardo del servidor, éste examina 1er byte del mensaje para ver quien es el cliente. Si la maquina es del mismo tipo que la del servidor, no necesita conversión. En caso contrario, el resguardo del servidor se encarga de la conversión.

También se puede acordar una representación a la que se adapten todos (formas canónicas). Otro problema a analizar es el pasaje de parámetros por referencia. Supongamos que uno de los parámetros de la llamada del cliente sea un apuntador a un arreglo (que está dentro del espacio de direcciones del proceso llamador).

Cuando el resguardo del cliente se hace cargo, sabiendo que ese parámetro apunta a un arreglo y conociendo su tamaño, puede decidir copiar todo el arreglo en el mensaje. Sun RPC usa un lenguaje de interfaz que se llama XDR y un compilador de interfaces que se llama rpcgen.

En un ambiente distribuido que utiliza RPC se pueden dar diferentes fallas, a saber:

1. El cliente no puede localizar el servidor
2. Se pierde el mensaje de solicitud del Cliente al servidor.

3. Se pierde el mensaje de respuesta del servidor al cliente.

4. Fallas en el servidor

En el caso 1, puede ocurrir que el cliente no se haya ejecutado por mucho tiempo y tener una versión vieja para la comunicación con el servidor y por eso la llamada no sea la adecuada para ubicar el servidor. También puede ser que el servidor esté inactivo.

Puede que retorne un código de error, pero debe analizarse como manejarlo, distinguiendo el error del resultado de retorno válido (si el procedimiento retorna -1 podemos interpretarlo como error, pero también puede ser el resultado del cálculo solicitado).

En el 2do caso, se puede poner un cronómetro que se inicie al mandar el mensaje. Si luego de un tiempo no hay respuesta p reconocimiento (ACK) se vuelve a mandar. Si debe repetirlo muchas veces el cliente puede interpretarlo como un caso 1, es decir, que no se localiza el servidor.

En el 3er caso, puede haber una solución semejante al 2, es decir, cronometrar. Pero a veces supone que se perdió una respuesta y en realidad ocurre que el servidor es muy lento, o no sabe si se perdió la solicitud o la respuesta, y no puede volver a mandarse sin consecuencias graves.

Por ejemplo, el caso de una transferencia de dinero. Supongamos que desde un cliente se quiere solicitar una transferencia de dinero a un servidor. El

servidor realizó la transferencia de dinero pero se perdió la respuesta al cliente. Si el cliente vuelve a hacer la solicitud de transferencia pensando que no se realizó su solicitud, se hará una segunda transferencia de dinero. Cuando una solicitud se puede repetir n veces sin que ocurran daños, se dice que es idempotente.

Esta situación se puede solucionar con el secuenciamiento del mensaje, es decir, asociando un número de secuencia al mensaje para que el servidor diferencie entre el original y la retransmisión (en este ultimo caso, el número de secuencia será mayor pues es posterior). Así, se puede rechazar una retransmisión si en realidad se llevó a cabo la operación.

Puede agregarse un bit en el mensaje para distinguir original de retransmisión (siempre se llevan a cabo las originales; las retransmisiones se analizan) En el 4to caso, se plantean las fallas del servidor.

En el servidor la secuencia de eventos es recepción, ejecución y respuesta. Las fallas deben analizarse según:

- Si ocurrió luego de ejecutar la solicitud del cliente y antes de la respuesta: si es así, hay que informar al cliente que se llevó a cabo la operación.
- Si la falla ocurrió antes de ejecutar la solicitud del cliente: Sólo exige que el cliente retransmita. El servidor debe indicar este tipo de falla.

Siempre está la alternativa de no hacer nada, es decir, cuando el servidor falla, el cliente no tiene ayuda. La RPC puede realizarse desde ninguna a un número grande de veces.

Hay diferentes formas de atender este tipo de problema. Es lo que se llama semánticas RPC en fallas del servidor. Semántica al menos una vez: se espera a que el servidor vuelva a arrancar y se intenta realizar la operación hasta poder mandar una respuesta al cliente. Garantiza que la RPC se realice al menos una vez (puede realizarse más veces). Es la que se usa normalmente en RPC.

Semántica a lo sumo una vez: se informa de la falla. Puede ser que no se haya realizado y si se realizó, será una sola vez. Lo ideal sería una semántica de exactamente una, pero no hay forma de garantizarla. Semántica de no garantizar nada: Fácil de implementar. El RPC puede realizarse una vez, muchas o ninguna.

3.2.4.1. Fallas en el cliente

El cliente envía una solicitud y falla antes de recibir la respuesta. A esta operación se la llama huérfana. Esto provoca varios problemas pues se bloquean archivos, se desperdician ciclos de CPU, etc. Para este problema se plantan 4 soluciones diferentes:

- Exterminación. El resguardo de cliente envía un RPC y crea una entrada en un registro (bitácora) indicando lo que va a hacer y este registro se mantiene en un disco o donde sobreviva a las fallas. Al reiniciar se analiza y se elimina el proceso huérfano (el realizado en el servidor). Las consecuencias de eliminar el huérfano pueden ser que los bloqueos no se liberen en realidad, y si el proceso huérfano creó otros procesos en otros servidores, estos también quedarán huérfanos y no son localizables.
- Reencarnación. Se divide el tiempo en "épocas" numeradas en forma secuencial. Cada vez que el cliente arranca se considera una nueva época y así es informado a todos los servidores. De esta manera se destruyen cálculos remotos de épocas anteriores o, si llega al cliente una respuesta a una petición de una época anterior, se rechaza.
- Reencarnación sutil. Cuando el servidor recibe un mensaje de "nueva época" analiza si tiene cómputos remotos generados por épocas anteriores. Si hay, se los manda al cliente para que él decida su eliminación.
- Por expiración. A cada RPC se le asigna un período de tiempo T para completarse. Si no le alcanza, debe pedir otro quantum. Debe asegurarse que aún el cliente está activo.

El servidor esperará un tiempo T para reiniciar, para asegurarse la eliminación de los huérfanos. El problema de esta solución es fijar el valor de T.

3.2.5. RMI

Supongamos que un objeto A, a nivel de aplicación invoca un método en el objeto remoto B (también a nivel de aplicación). Para ello es necesario que A tenga una referencia remota a B. De ahora en más, A está en el proceso cliente y B en el proceso servidor.

En el cliente debe haber un *proxy* de B. El proxy, considerado parte del software RMI, tiene la función que la invocación al método remoto sea transparente para los clientes, comportándose como un objeto local y mandándole, en realidad, un mensaje de un objeto remoto.

El proxy se encarga de la referencia al objeto remoto, empaquetado de argumentos, desempaquetado de resultados y envío y recepción de los mensajes ante el cliente. Cada objeto remoto del que el cliente tenga una referencia, tiene un proxy.

El proxy llama al *módulo de referencia remota*. Cuando se empaquetan y desempaquetan las referencias a objetos remotos. Este módulo traduce las referencias a objetos locales y remotos y crea referencias. Hay un módulo por proceso. Cada módulo tiene una tabla sobre los objetos locales y remotos.

El objeto remoto B, estará en la tabla del servidor. Pero su proxy, que está en el proceso local, figura en la tabla del cliente. Cuando se pasa un objeto remoto por primera vez, se le pide a este módulo que cree la referencia remota y que lo añada a su tabla.

Si se invoca a un objeto remoto que no está en la tabla del módulo, se le pide al software RMI que le cree un proxy. En el servidor, está el resto del software RMI: el distribuidor y el esqueleto para la clase del objeto B.

El distribuidor recibe el mensaje de petición que llega desde el proceso cliente, selecciona el método apropiado del esqueleto y la pasa el mensaje de petición. El esqueleto implementa los métodos de la interfaz remota. El método desempaqueta los argumentos del mensaje e invoca al método correspondiente.

Luego de ejecutada la invocación, empaqueta el resultado para que sea enviado al proxy del objeto en el cliente en un mensaje de respuesta. Para la comunicación entre el proceso cliente y el servidor, hay módulos de comunicación de ambos lados.

RPC fue siempre una herramienta importante para explicar el contexto de conectividad en sistemas distribuidos. Hoy no es considerado middleware, pues RPC requiere que los programadores reescriban sus aplicaciones una y otra vez a medida que las aplicaciones cambian o se multiplican. Y esto no es coherente con la simplicidad que enuncia en su definición.

No obstante en muchos libros de Sistemas Distribuidos (Colouris), se sigue considerando parte del middleware. Por ejemplo, MOM, al contrario que RPC, es una forma de comunicación asincrónica, donde el sender no hace Block waiting a la espera de respuesta. Con las definiciones vertidas en el

marco teórico, puede interpretarse erróneamente que el middleware sólo permite trabajar con ambientes totalmente distribuidos.

Es importante considerar que los servicios que se quieran ofrecer descansan sobre una infraestructura, donde una parte puede requerir centralización, y otras pueden ser mantenidas por otros departamentos.

Por lo tanto, un objetivo importante es no perder de vista que a veces se hace necesario crear un servicio centralizado que provea autorizaciones, herramientas, para poder implementar la distribución.

El middleware debe proporcionar servicios para que sean usados por parte de las aplicaciones: servicios para gestión de nombres, seguridad, notificación de eventos, almacenamiento persistente, etc.

3.2.6. NET Framework Remoting

El .NET Remoting es una tecnología de objetos distribuidos sucesora de DCOM.

La infraestructura de .NET Remoting es un enfoque abstracto de la comunicación entre procesos. La mayor parte del sistema funciona sin llamar la atención. Por ejemplo, los objetos que se pueden pasar por valor, o copiar, se pasan automáticamente de una aplicación a otra en dominios de aplicación o en equipos distintos.

Pero la verdadera ventaja del .NET Remoting es su capacidad para permitir la comunicación entre objetos pertenecientes a dominios de aplicación o a procesos distintos mediante diferentes protocolos de transporte, formatos de serialización, esquemas de duración de objetos y modos de creación de objetos. Además, la interacción remota permite intervenir en prácticamente todas las fases del proceso de comunicación, sea cual sea la razón.

Tanto si se implementa una serie de aplicaciones distribuidas como si sólo se desea mover componentes a otros equipos para aumentar la escalabilidad del programa, resulta más fácil si se considera el .NET Remoting como un sistema genérico de comunicación entre procesos con algunas implementaciones predeterminadas capaces de controlar fácilmente la mayoría de los escenarios posibles.

La comunicación entre procesos requiere un objeto servidor cuya funcionalidad esté a disposición de los que realizan las llamadas fuera de su proceso, un cliente que realice llamadas al objeto servidor y un mecanismo de transporte que lleve las llamadas de un extremo a otro. Las direcciones de los métodos del servidor son lógicas y funcionan correctamente en un proceso, pero no funcionan en otro proceso de cliente. Para solucionar este problema, el cliente puede llamar a un objeto servidor realizando una copia de todo el objeto y pasándola al proceso de cliente, donde se pueden invocar directamente los métodos de la copia.

No obstante, muchos objetos no se pueden o no se deben copiar ni pasar a otros procesos para ejecutarse. Los objetos sumamente grandes con muchos métodos son los menos aconsejables para copiar o pasar por valor a

otros procesos. Normalmente, un cliente sólo necesita la información devuelta por un solo método o por unos pocos en el objeto servidor. Copiar el objeto servidor entero, incluyendo lo que podrían ser enormes cantidades de información interna o estructuras ejecutables no relacionadas con las necesidades del cliente, supondría desperdiciar el ancho de banda así como la memoria del cliente y el tiempo que se emplea en procesarlo todo. Además, muchos objetos exponen una funcionalidad pública pero requieren datos privados para la ejecución interna. Copiar estos objetos podría permitir que clientes no autorizados examinaran datos internos, lo que posibilitaría la aparición de problemas de seguridad. Por ultimo, algunos objetos utilizan datos que no se pueden copiar de ninguna forma comprensible.

En estos casos, el proceso del servidor debería pasar al proceso del cliente una referencia al objeto servidor, en lugar de una copia del objeto. Los clientes pueden utilizar esta referencia para llamar al objeto del servidor. Estas llamadas no se ejecutan en el proceso del cliente. En vez de eso, el .NET Remoting reúne toda la información referente a la llamada y la envía al proceso del servidor, donde se interpreta, se busca el objeto del servidor correcto y se realiza la llamada a dicho objeto en nombre del objeto del cliente. A continuación, se devuelve el resultado de la llamada al proceso del cliente para que se lo devuelva a su vez al cliente. El ancho de banda se utiliza únicamente para la información esencial: la llamada, sus argumentos y todas las excepciones o valores devueltos.

3.2.6.1. Arquitectura simplificada de interacción remota

El uso de referencias a objetos para la comunicación entre objetos de servidor y clientes es la esencia de la interacción remota. No obstante, la arquitectura de interacción remota proporciona al programador un procedimiento aún más sencillo. Si configura correctamente el cliente, sólo tiene que crear una nueva instancia del objeto remoto mediante new (o la función de creación de instancias del lenguaje de programación administrado que utilice). Su cliente recibe una referencia al objeto de servidor, lo que le permite llamar a sus métodos como si el objeto estuviera en su proceso en lugar de estar ejecutándose en otro equipo. El sistema de interacción remota utiliza objetos proxy para dar la impresión de que el objeto del servidor se encuentra en el proceso del cliente. Los objetos proxy son objetos complementarios, que se presentan como si fueran otro objeto. Cuando un cliente crea una instancia del tipo remoto, la infraestructura de interacción remota crea un objeto proxy que, para su cliente, tiene exactamente la misma apariencia que el tipo remoto. Su cliente llama a un método en ese objeto proxy y el sistema de interacción remota recibe la llamada, la dirige hacia el proceso del servidor, invoca al objeto de servidor y envía el valor devuelto al objeto proxy del cliente, que a su vez devuelve el resultado al cliente.

Las llamadas remotas deben ser transmitidas de alguna forma entre el cliente y el proceso del servidor. Si se crea un sistema de interacción remota, se podría empezar por la programación de redes y una amplia gama de protocolos y especificaciones de formatos de serialización. En el sistema .NET Remoting, la combinación de tecnologías subyacentes necesarias para abrir

una conexión de red y utilizar un determinado protocolo para enviar los bytes a la aplicación receptora se representa como un canal de transporte.

Un canal es un tipo que toma una secuencia de datos, crea un paquete según un determinado protocolo de red y lo envía a otro equipo. Algunos canales sólo pueden recibir información, otros sólo pueden enviarla y otros, como las clases predeterminadas del .NET Remoting, TcpChannel y HttpChannel, los que se pueden utilizar en ambos sentidos.

Aunque el proceso del servidor conoce perfectamente todos los tipos, el cliente sólo sabe que necesita una referencia a un objeto de otro dominio de aplicación, puede que de otro equipo. Desde el exterior del dominio de aplicación de servidor, una dirección URL ubica el objeto. Las direcciones URL que representan tipos únicos para el mundo exterior son direcciones URL de activación, que garantizan que su llamada remota va dirigida al tipo apropiado.

3.2.6.2. Diseño completo de un sistema de interacción remota

Imaginemos que se ejecuta una aplicación y se desea utilizar la funcionalidad expuesta por un tipo almacenado en otro equipo. En la figura siguiente se muestra el proceso general de interacción remota.

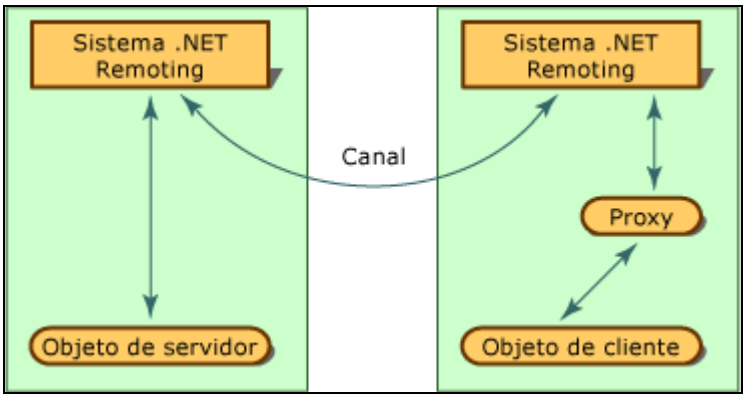

Figura 3.4. Proceso de interacción remota [MOL 2004]

Si ambas partes de la relación están configurados correctamente, un cliente se limita a crear una nueva instancia de la clase de servidor. El sistema de interacción remota crea un objeto proxy que representa a la clase y devuelve al objeto del cliente una referencia al objeto proxy. Cuando un cliente llama a un método, la infraestructura de interacción remota controla la llamada, comprueba el tipo de información y dirige la llamada por el canal hacia el proceso del servidor. Un canal a la escucha detecta la solicitud y la reenvía al sistema de interacción remota del servidor, que a su vez busca (o crea, si es necesario) y llama al objeto solicitado. A continuación el proceso se invierte: el sistema de interacción remota del servidor incluye la respuesta en un mensaje que el canal del servidor envía al canal del cliente. Por último, el sistema de interacción remota del cliente devuelve el resultado de la llamada al objeto del cliente a través del objeto proxy.

Para que esto funcione se necesita muy poco código propiamente dicho, pero es conveniente reflexionar un poco sobre el diseño y la configuración de la relación. El código puede ser totalmente correcto y aún así no funcionar porque una dirección URL o un número de puerto no lo sean.

Aunque esta información general del proceso básico del .NET Remoting es bastante sencilla, los detalles de los niveles inferiores pueden resultar muy complejos.

3.2.7. Servicios Web (WS)

Los Servicios Web, permiten la interacción entre servidores, servidores y clientes, y clientes entre sí.

A medida que crece la necesidad de comunicar aplicaciones para lograr la interoperabilidad, se va haciendo más necesario el uso de Servicios Web, que provean un medio de comunicación estándar entre diferentes aplicaciones de software que corren en distintas plataformas y frameworks.

3.2.7.1. Conceptos y ventajas

Los Servicios Web pueden verse más como una evolución en el campo de los sistemas distribuidos que como una revolución. El uso de aplicaciones XML ha sido un paso definitorio para esta evolución.

Según Graham Glass, Servicios Web es *una colección de funciones que están empaquetadas en una entidad simple y ofrecida en la Red para que otros programas la usen. Son building blocks para crear sistemas distribuidos y*

permitir que empresas y usuarios en general puedan lograr, de una forma rápida y barata, su inserción en la Web.

El objetivo es desarrollar aplicaciones distribuidas altamente integradas que interactúen por XML entre los servicios Web y múltiples servicios WEB. El uso de XML, que será analizado más adelante, se prioriza en este modelo por permitir la transmisión de información auto descriptiva a una plataforma neutral.

Supongamos que un usuario visita un sitio Web, y quiere transmitir información directamente a una aplicación cliente, para su proceso, lo cual hoy se hace, la mayoría de las veces, con la intervención del usuario. El objetivo es que las aplicaciones en Internet puedan comunicarse entre sí. El usuario podría obtener un resultado del procesamiento de información proveniente de distintos sitios.

Otra situación ilustrativa es que el usuario pueda usar distintos dispositivos cliente, con resoluciones de pantalla diferentes, o utilizar un nuevo dispositivo. De manera transparente para el usuario, la aplicación en el servidor, generará XML, y la transformación necesaria para el dispositivo particular, lo cual se puede hacer en el servidor o en el cliente. De allí la necesidad que todos los dispositivos tengan un soporte para XML, para implementar esta interacción.

En caso de utilizar un nuevo dispositivo, éste se autentica, y elegirá cuáles servicios usará para manejar la información y recuperarla.

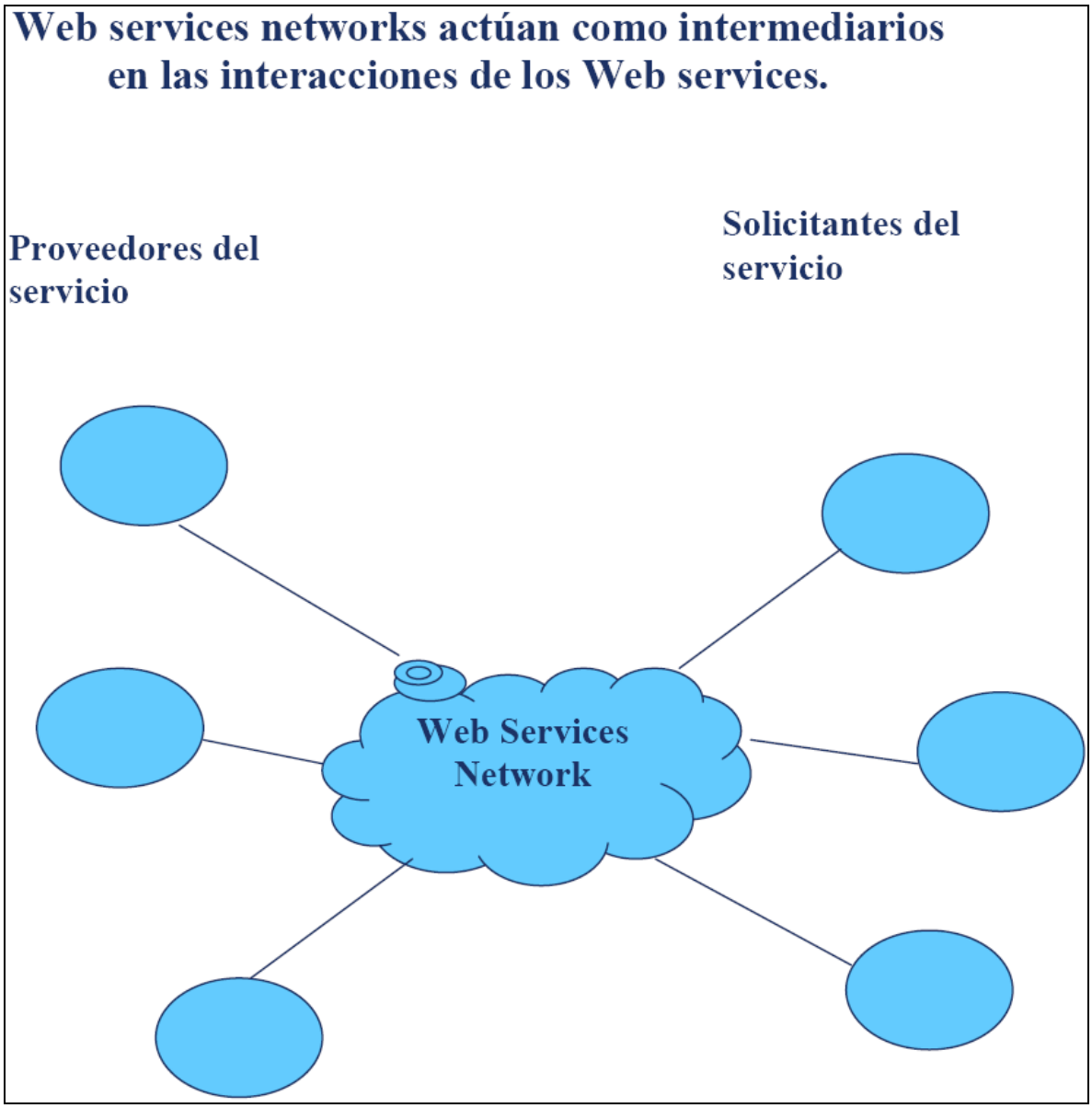

Figura 3.5. Funcionamiento de los Servicios Web [MOL 2004]

Veamos por ejemplo, como evoluciona la distribución del software. Hasta no hace mucho tiempo, la única posibilidad era distribuir las aplicaciones en CD's con actualizaciones periódicas. La alternativa de cambio que estamos analizando debe ofrecer una conexión directa con el proveedor para la entrega de actualizaciones, soporta, etc. En fin: servicios. El cambio debe estar orientado al servicio a través de la Red.

Analicemos ahora la relación del usuario con las interfases gráficas. Este modelo debe permitir que la interfase se presente al usuario según una personalización, basada en la identidad. Ante la necesidad de parte del usuario de agregar reconocimiento de voz, no sería necesaria la adaptación de la aplicación para, por ejemplo, un usuario con algún tipo de discapacidad visual o auditiva.

Si el usuario se autentica en algún site, deberían viajar las personalizaciones propias que regirán la relación con otros clientes o servidores. Por lo tanto, con estas pautas, se deduce la necesidad de usar agentes inteligentes, para que en los servicios Web se pueda trabajar sobre el tipo de información que el cliente quiere recibir.

Para lograr la implementación del modelo es necesario el avance tanto del HW como del SW para soportar los protocolos de alto nivel. Los servidores para implementar este modelo deben permitir la escalabilidad desde el SO para mostrar un conjunto de servidores como un único servidor lógico.

Los beneficios de utilizar WS se manifiestan en:

- Usa estándares abiertos, basados en texto. Se pueden comunicar componentes escritos en diferentes lenguajes y distintas plataformas.
- Fácil de implementar. No es costoso: se usa una infraestructura existente.
- La mayoría de las aplicaciones pueden re-empaquetarse

3.2.7.2. Arquitectura de los Servicios Web

La Arquitectura básica de Servicios Web debe incluir lo necesario para:

- El intercambio de mensajes entre las entidades
- Publicar en la Red, poner en conocimiento de la existencia de ese Web
- Service, para que pueda ser utilizado por algún cliente
- Encontrar otros Servicios Web

3.2.7.3. Dinámica de los WS Protocolos

Una de las partes interesantes (y fundamentales) de Servicios Web es la combinación de protocolos estándar sobre Internet. Se enfatiza la necesidad de usar estándares para el intercambio de datos entre aplicaciones de distintas plataformas y lenguajes.

¿Porqué hablar de SOAP, XML, WSDL? ¿Porqué no seguir con COM, o CORBA? Básicamente es el modelo de interoperabilidad de plataformas y aplicaciones lo que exige optar por estos protocolos. COM (de Microsoft), y OMG CORBA no interoperan, y no han demostrado la extensibilidad que se quiere lograr sobre la Red.

El uso de protocolos estándar es necesario para lograr la Interoperabilidad en ambiente heterogéneos, con Independencia de SO, Lenguaje, versiones.

- HTTP en transporte
- XML para codificar los datos
- WSDL para describir el servicio
- SOAP para invocar llamados remotos y “envolver” el intercambio
- UDDI, para publicarlos (serían las páginas amarillas de servicios Web)

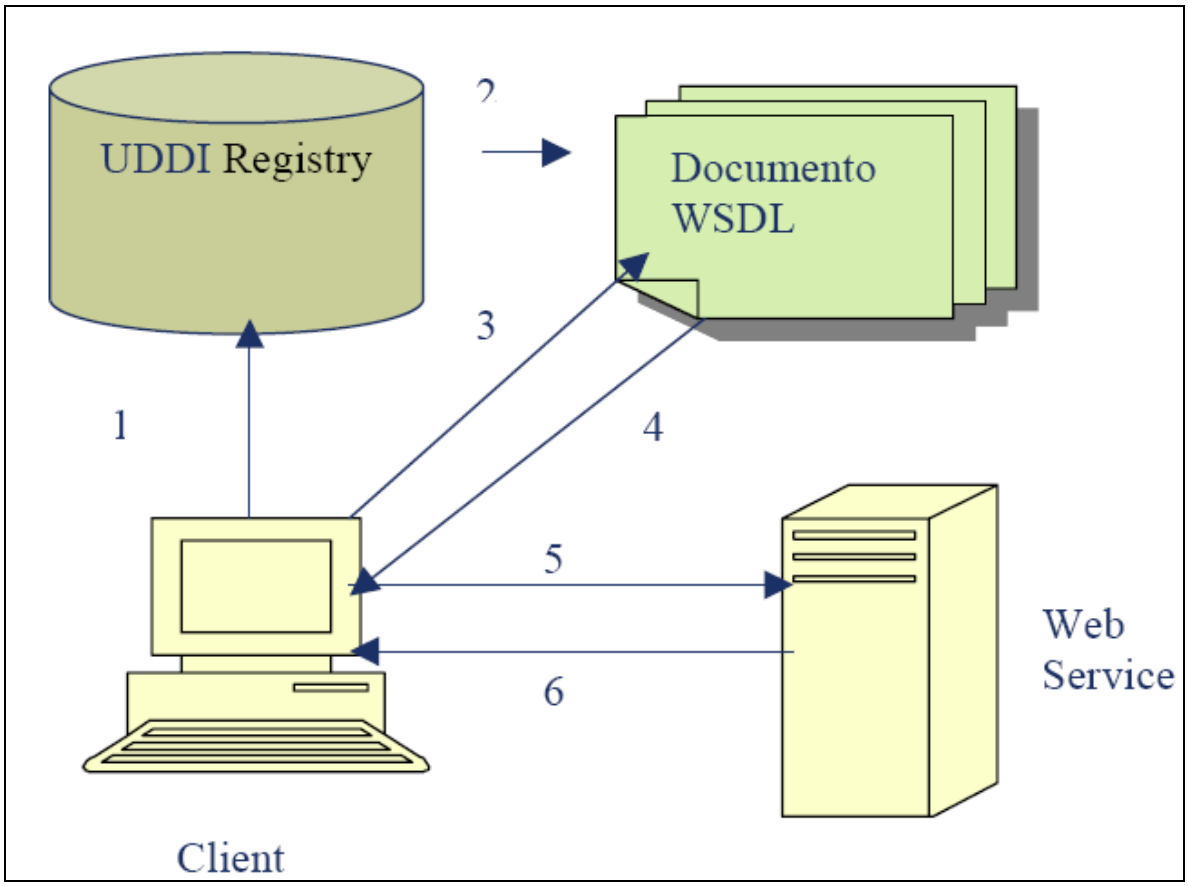

Figura 3.6. Interacción de los Estándares de los Servicios Web [MOL 2004]

1. Cliente pregunta al registry para ubicar un servicio.
2. Registry le indica al cliente un documento WSDL.
3. Cliente accede al documento WSDL.
4. WSDL provee lo necesario para interactuar con Servicio Web.
5. Client envía un requerimiento usando SOAP.
6. Servicio Web retorna una respuesta SOAP.

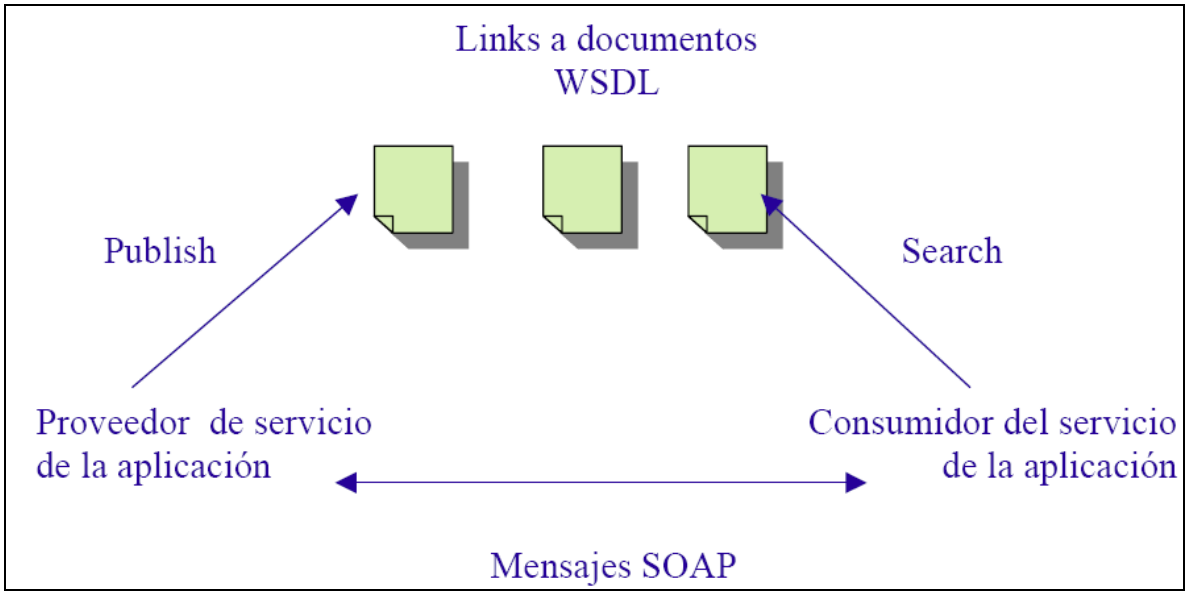

Figura 3.7. Interacción de los Estándares de los Servicios Web (detalle) [MOL 2004]

La interacción se da a través de un patrón de mensajes (message exchange patterns, MEPs) que definen la secuencia de intercambio.

Se debe definir una descripción del Servicio Web, que indique su funcionalidad. Descripción incluye: como invocar el Servicio Web y qué retorna. Involucra tipos de datos, estructura, MEP's, y la dirección del proveedor del servicio.

En esta arquitectura hay tres roles bien definidos:

- Proveedor del servicio
- Lugar o guía para la búsqueda de servicios
- Solicitante del servicio

Uno de los pilares de esta tecnología es la reusabilidad del software. Por lo tanto, el desarrollador “publica” el Servicio Web desarrollado para que esté disponible para el cliente que lo necesita.

Para ello, se debe “registrar” el Servicio Web en un lugar que se llama ***service discovery agency***, done el proveedor “hostea” el módulo de software para que sea accesible desde la Red.

La interacción se da a través de un patrón de mensajes, **MEP**s, que definen la secuencia de intercambio. El proveedor debe además, definir una descripción del Servicio Web para que el cliente analice si realmente otorga la funcionalidad que necesita. En esa descripción también explica como invocar el Servicio Web y qué retorna, por lo tanto, involucra tipos de datos, estructura, MEPs, y la dirección del proveedor del servicio.

Un agente en este tipo de arquitectura, puede ser proveedor, solicitante o cumplir ambos roles, discovery agency y los otros dos. Cuando el solicitante “descubre” el servicio en la agency, lo puede invocar de acuerdo a la descripción ofrecida, para que se cumpla el binding.

Lo interesante de tecnologías que usan XML, SOAP, es que se pueden integrar en aplicaciones ya existentes sin demasiada complejidad.

Productos que usan SOAP pueden usar servidores HTTP existentes y procesadores XML que tenga el cliente.

Si, en cambio, se utiliza CORBA, DCOM o Java RMI, se debe instalar el ambiente apropiado, configurar el sistema para adaptarlo a la nueva infraestructura distribuida.

También puede exigir reconfigurar firewalls. De allí que a estos sistemas se los llama “heavyweight systems”. La computación ubicua requiere un conjunto de protocolos disponibles y listos para ser usados, que implemente:

- Un mecanismo de serialización que transforme la llamada al método en la forma adecuada para su transmisión sobre la red
- Una capa de transporte que lleve esos datos inherentes al método entre los sistemas remotos
- Un mecanismo para encontrar, activar y desactivar objetos distribuidos
- Un modelo de seguridad para proteger el sistema local y remoto.

Cuando el objetivo es la interoperabilidad, la elección de los protocolos debe ser cuidadosa. Por ejemplo, sistemas distribuidos de objetos como COM, de Microsoft, y OMG CORBA son estándares que no interoperan.

Por lo tanto, se hace imprescindible adaptarse a los estándares. Usar XML no garantiza por si sólo la interoperabilidad: la combinación del conjunto de protocolos que se utilicen, son los que me garantizarán la interoperabilidad deseada.

La tecnología SOAP y WSDL se desarrollaron en principio fuera de W3C, pero sus sucesores se trabajaron dentro del consorcio. La especificación

SOAP 1.2 está siendo usada como la base para crear un mesagging framework escalable, y WSDL 2.0 se usa como el lenguaje de definición de interfases.

3.2.7.4. Operaciones

En WS, las operaciones básicas son publicar (publish), encontrar (find) e interactuar (interact).

- **Publish**: el desarrollador publica su WS para ofrecerlo a la comunidad de potenciales usuarios del servicio. Lo que se publica es la **descripción del servicio**.
- **Find**: a través de esta operación, un desarrollador busca un servicio directamente o haciendo queries al registro de servicios.
- La operación se invoca tanto cuando se busca la descripción del servicio, como cuando se desea acceder al servicio en runtime (*binding invocation)*.
- **Interact**: en tiempo de ejecución (runtime) el que requiere el servicio inicia una interacción al invocar el WS, haciendo uso de la descripción para ubicación, contacto e invocación del servicio.

  La interacción puede ser sincrónica o asincrónica. El modelo puede ser desde broadcast a varios servicios, procesos de negocios (business process), conversación multilenguaje, etc.

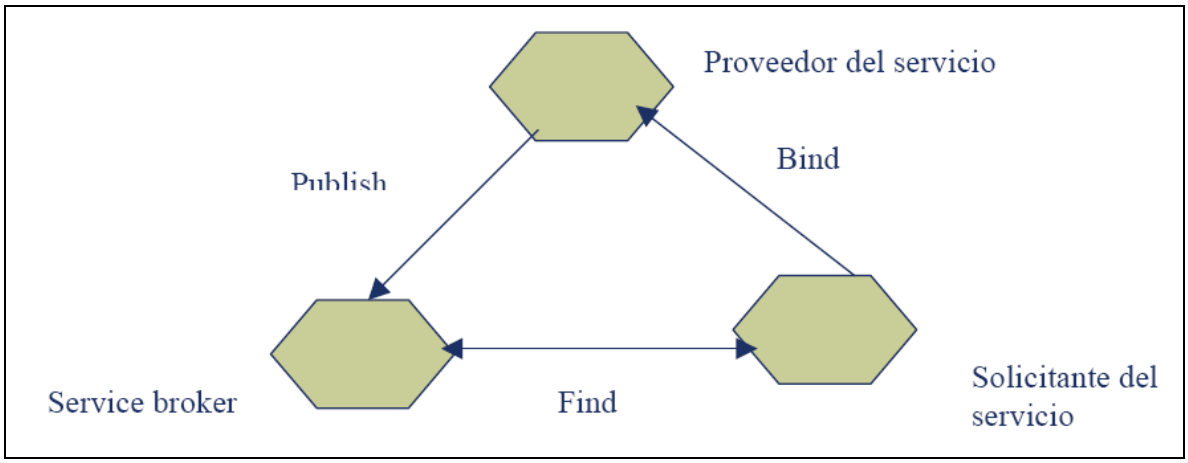

Figura 3.8. Operaciones en los Servicios Web [MOL 2004]

3.2.7.5. Servicios Web estático y dinámico

En los estáticos, el cliente invoca a un proveedor fijo y ubica el archivo WSDL a través de mail, FTP, ó registro UDDI. Luego, la aplicación cliente, invoca al proveedor de WS.

En los dinámicos, no se sabe a qué proveedor se va a terminar invocando. La aplicación cliente interactuará con el registro UDDI a través de API's. Recupera la información desde el registry y busca a los proveedores que ofrecen ese servicio y entonces, invoca.

3.2.7.6. WS-I: Servicios Web y estándares

WS-I es una organización que promueve la interoperabilidad entre plataformas y lenguajes de programación usando WS. Hace recomendaciones y provee una guía de trabajo.

Fue fundada para crear y promover el uso de protocolos genéricos para el intercambio de mensajes entre servicios.

Protocolos genéricos son protocolos que son independientes de cualquier acción específica que no sea la seguridad, confiabilidad o entrega eficiente del mensaje.

El WS-I Basic Profile 1.2 es un conjunto de especificaciones de WS no propietarios, junto con aclaraciones y recomendaciones para promover la interoperabilidad.

3.2.7.7. El proceso de negocio en WS

Servicios Web permite la interoperabilidad entre aplicaciones. Hasta ahora se ha definido el espacio de los WS a través de SOAP, UDDI, WDSL, XML.

Pero el objetivo es la interacción de sistemas heterogéneos en el sector Salud. Y hasta aquí hemos visto WS como la forma de integrar sistemas a través de la interacción simple de protocolos estándar. No hemos incorporado al proceso de negocio como parte de este modelo. Sin él, no hay verdadera integración.

Los negocios se benefician con el efecto network. Para ejemplificar este efecto se utiliza el caso de la máquina de Fax: la primera, única en el mercado, no tenía ningún beneficio. Cuando 100 personas tuvieron máquina de fax, empezó a considerarse productiva, pues permitía la interacción de esos 100 usuarios. Cuando se extendió a millones, pasamos hablar de una poderosa herramienta para el crecimiento del negocio.

Lo mismo ocurre con las aplicaciones. Si cada vez más usuarios se conectan a ellas, facilitando la conexión, se amplía la posibilidad de negocio. IT (Information Technology) necesita cada más interoperabilidad, a menor costo, y con un mecanismo de negocio soportado por la mayor cantidad de empresas para trabajar juntos.

Las relaciones de negocio, por ejemplo entre un cliente y un proveedor, se componen de diferentes partes, unas públicas, otras privadas. Una lista estimativa de precios puede ser pública, también los servicios que se ofrecen. Pero el cálculo del precio final o qué se asume como costo a incorporar al producto, o la justificación de un rechazo de pedido, puede desearse mantener en privado.

Por ello, en el proceso de negocio se habla de partes “visibles” de la interacción (la pública) y otras “no visible” (la privada). Otra razón por la que puede ser importante separar estas partes es que los cambios en los aspectos privados de la implementación de negocio (por ejemplo, una nueva forma de fabricar un producto, o el cambio de las relaciones con los proveedores de la materia prima) no afecten la parte pública.

3.2.7.8. Transparencia y opacidad

Los datos transparentes (de uso público o externo) afectan al protocolo de manera directa. Los opacos (de uso privado o interno) lo afectan de manera no determinística.

Por ejemplo, en la relación vendedor-comprador, el vendedor ofrece un servicio. El comprador libera una orden de compra que el vendedor acepta o rechaza. Esta decisión se basa en un criterio, pero este proceso de decisión es "opaco". ¿Cómo refleja esta opacidad en el protocolo externo? Enumerando alternativas que se presentarán como el resultado: el vendedor tiene derecho a reservarse el criterio por el cual acepta o rechaza una orden de compra

3.2.7.9. Niveles en el acuerdo de partes

Se acuerdan tres niveles entre las partes del negocio: nivel de transporte, de documentos de negocio y proceso de negocio.

El nivel de transporte es responsable por mover los mensajes o documentos entre una parte y la otra, de una manera segura, confiable y consistente. Incluye el aspecto transaccional (considerando que hablamos de aplicaciones distribuidas).

El nivel de documento se enfoca hacia el contenido del mensaje. El documento debe ser entendido por ambas partes.

El tercer nivel es el nivel de proceso de negocio. Define la interacción, los procesos internos y externos de ambas partes. Considerando el efecto network citado anteriormente, es importante que la mayor cantidad de nodos soporten los mismos protocolos de la capa de transporte. Para cumplir este objetivo, en estos protocolos se den cumplir estos cuatro requerimientos:

1. Deben ser neutrales en cuanto a lenguaje, sistema operativo, modelo de bases de datos y arquitectura.

2. Deben ser modulares y “componibles”.

3. Deben ser de propósito general adecuados para la integración de los distintos ambientes (BC, B2B, procesos internos, etc.).

4. Deben permitir la extensibilidad de las aplicaciones.

3.2.7.10. Conceptos asociados al proceso de negocio

Se llaman procesos ejecutables de negocio (Executable Business Processes) a los procesos que modelan el comportamiento real de las partes que participan en una interacción.

Los Protocolos de negocios (Business Protocols) son protocolos que especifican el intercambio de mensajes de las partes que intervienen (parte visible). No se ocupan del comportamiento interno (parte “no visible”). Las descripciones de estos procesos que manejan estos protocolos se llaman procesos abstractos (abstract processes).

Una forma de modelar los procesos ejecutables y los abstractos, puede implementarse a través de BPEL4WS. Es un lenguaje que permite especificar formalmente los procesos de negocio y la interacción de las partes, usando los protocolos estándar. Esta funcionalidad extiende la interoperabilidad de los WS, al incorporar la formalización de los procesos de negocio.

3.2.7.11. XML

XML proviene de eXtensible Markup Language, es decir, lenguaje extensible de marcado. Fue desarrollado por W3C para superar las limitaciones de HTML.

La ventaja de HTML está dada por que es soportado por la mayoría de las aplicaciones. Ocurre con el software de correo electrónico, exploradores, editores, bases de datos, etc.

Pero HTML fue creciendo desde su primera versión. Hoy la gran cantidad de etiquetas y sus combinaciones, sumado a nuevas tecnologías de soporte (JavaScript, Flash, CGI, ASP, etc.), y las aplicaciones van necesitando aun más etiquetas.

El advenimiento de pequeños clientes menos poderosos que las PC's no permiten procesar un lenguaje de acceso a la Web muy complicado. Por ello se desarrolló XML. Se basó en HTML, dada su popularidad, su éxito. Pero XML vino para cumplir con nuevas demandas.

Hay dos clases de aplicaciones XML: la publicación electrónica y el intercambio de datos.

Los dos cambios esenciales que XML hace a HTML es no predefinir etiquetas y ser estricto.

Veamos un ejemplo como definiríamos en XML una etiqueta para precio:

<precio moneda=”pesos”>850.00</precio>

La X de XML proviene de eXtensible, porque no predefine etiquetas, pero le permite al usuario crear las que precise para su aplicación.

Cómo sabe el explorador que una definición XML equivale a un conjunto de etiquetas en HTML? Es a través de las hojas de estilo. XML es compatible con la generación actual de exploradores.

Otra ventaja, que, desde el desarrollador puede verse como no tan beneficiosa, es la sintaxis estricta de XML. HTML no la tiene. La falta de dicha exigencia, si bien es cómoda para el desarrollador, requiere exploradores cada vez más complejos.

Al tener una sintaxis estricta XML puede ser usado en exploradores simples como exigen los dispositivos de mano actuales. A los archivos XML se les llama documentos. Esto hace que pueda confundirse el uso de XML y pensarlo sólo como una solución para la publicación electrónica.

3.2.7.11.1. Estructura de un documento XML

En general, en un documento podemos distinguir:

Título

Encabezado, donde se detalla remitente, destinatario, asunto

Texto, con párrafos, donde pueden incluirse URL’s, firmas.

3.2.7.11.2. Algunas definiciones

- **Marcado**

En la publicación tradicional es una actividad independiente que se realiza después de la escritura de un documento y antes de la composición de las páginas. Estas indicaciones guían al tipógrafo en cuanto a la apariencia del documento (fuentes, destacados, etc.).

El documento electrónico es el código que se inserta en el texto del documento que almacena la información necesaria para el procesamiento electrónico. En el caso de un documento XML, el marcado contribuye a identificar la estructura.

- **Marcado procedural**

En el procesamiento de texto, el usuario especifica la apariencia del texto (negrita, párrafo en cursiva, fuente, o una posición particular del texto dentro de la página).

Esto se almacena como código dentro del texto. Se le llama marcado procedural pues deriva en un procedimiento para un dispositivo de salida, por ejemplo, la impresora. El RTF, Formato de Texto Enriquecido, es un marcado procedural desarrollado por Microsoft, y está soportado por la mayoría de los procesadores de texto.

No obstante sus ventajas, adolece de algunos problemas:

1. No almacena información sobre estructura.

2. No es flexible (cambios en las normas de formato, exigen cambios en el documento).

3. Es un proceso lento y propenso a errores.

- **Codificación genérica**

En la codificación genérica se utilizan macros que pueden mejorar el marcado procedural. En vez de implementar controles, se puede invocar procedimientos externos de formato. La etiqueta (o identificador genérico) se agrega al elemento de texto. Estas etiquetas están asociadas a normas de formato.

Tex es un ejemplo de codificación genérica.

Los beneficios con respecto al marcado son:

1. Hay mayor portabilidad y flexibilidad

2. Se graba información sobre estructura.

- **SGML**

Extiende la codificación genérica al agregar descripción sobre estructura del documento y se ajusta a un modelo, lo que permite ser procesado por software o guardarlo en una base de datos.

A través de SGML empieza la corriente por la cual los autores cuentan con un lenguaje para descubrir la estructura de sus documentos y los marquen.

La gran diferencia entre la codificación genérica y el SGML es que el marcado describe la estructura del documento.

La estructura del documento se describe en una DTD (data type definition) que también recibe el nombre de aplicación SGML. En esta DTD se especifican elementos, relaciones y etiquetas que definen una estructura de documento y lo marcan.

En SGML el marcado sigue a un modelo. Algunos lenguajes se basan en SGML como es el caso de HTML (lenguaje de marcado para documentos WEB), CALS (estándar para intercambio de documentos del DoD, Department of Defense of EEUU), DocBook, etc.

- **HTML**

Es la aplicación más popular de SGML. Es un conjunto de etiquetas que siguen la norma SGML. HTML ha evolucionado, considerando que:

  - Integra etiquetas de formato logrando características de marcado procedural

  - Agrega el atributo class y hojas de estilo, logrando características de codificación genérica.

3.2.7.11.3. Las aplicaciones XML. Documentos y datos.

Las aplicaciones XML se clasifican en:

- Aplicaciones orientadas a documento, información dirigida al consumo humano;
- Aplicaciones de datos que manejan información dirigida al software. Es una diferencia cualitativa. El estándar es el mismo.

3.2.7.11.3.1. Aplicaciones de documento

Veamos la publicación de un documento que quiere imprimirse, mostrarse en la WEB y llevarlo a una palm. XML me da la posibilidad de publicarlos automáticamente en distintos medios.

3.2.7.11.3.2. Aplicaciones de datos

Si la estructura de un documento puede expresarse como XML, también la estructura de una base de datos.

Veamos como incluiríamos una lista de productos con su precio en XML.

<?xml version="1.0" enconding ="ISO-8859-1"?>

<productos>

<id producto="p1">

<nombre>Libro de Silverschatz</nombre>

<precio>140</precio>

</producto>

<id producto="p2">

<nombre>Libro de Tenenbaum</nombre>

<precio>150</precio>

</producto>

<id producto="p3">

<nombre>Libro de Stallings</nombre>

<precio>180</precio>

</producto>

</productos>

3.2.7.12. Servicios Web usando XML (XML Servicios Web)

XML Servicios Web fue diseñado para que sistemas heterogéneos puedan interactuar, comunicándose e intercambiando información. En el marco que nos ocupa, la heterogeneidad está orientada a sistema operativo, lenguaje o arquitectura.

Para esta interacción, XML Servicio Web usan HTTP, XML y SOAP, es decir, estándares en el mundo Internet, lo cual es coherente con el objetivo de interoperabilidad. De esta manera, el servicio es accesible desde cualquier cliente.

Estos servicios pueden verse como caja negras (black boxes). La aplicación que los invoca se abstrae de la funcionalidad: la aplicación necesita saber qué hacen, no cómo lo hacen.

Para poder accederlos deben saber cómo invocarlos y qué retornará de la ejecución (que, obviamente, será remota). En otras palabras: los desarrolladores crean métodos y los exponen para que otros los usen.

Veamos tres protocolos que rigen este intercambio entre el cliente y el XML Servicio Web.

- HTTP GET: el XML Servicio Web es invocado usando un requerimiento GET sobre HTTP. El input se pone en un string (query string) y el resultado vuelve como un documento XML.

- HTTP POST: el XML Servicio Web es invocado usando un requerimiento POST sobre HTTP. Los valores de input se ponen en el cuerpo del HTTP POST y el resultado vuelve como un documento XML.
- SOAP: el XML Servicio Web es invocado usando un mensaje SOAP sobre HTTP. Los valores de input se ponen en el cuerpo del mensaje SOAP y el resultado vuelve como un documento XML.

3.2.7.12.1. Exposición del servicio al cliente

El usuario que desarrolló el servicio, necesita exponerlo, para que aquellos usuarios interesados puedan accederlos.

¿Cómo publicita el desarrollador su servicio? Si bien siempre se puede hacer llegar al desarrollador interesado directamente la URL del lugar donde está el servicio, existe una forma más práctica: registrar el servicio en un site, que funciona como una guía de los servicios en la Red, una gran base de datos de servicios. Esta es la función de la base de datos UDDI (Universal description, Discovery and Integration).

UDDI es una organización para el registro, esencialmente, de XML Servicio Web. De esta manera, potenciales clientes que precisan determinado servicio, lo pueden buscar en UDDI, y si existe, saben cómo accederlo y dónde.

3.2.7.12.2. Invocación del servicio por el cliente

Cuando se crea el XML Servicio Web, se debe considerar que el cliente que lo necesite lo invocará. Para ello, debe informarse cómo hacerlo, cuáles son los métodos que lo forman, cuáles serán los argumentos de llamada y qué devolverá.

Para eso se agrega al XML Servicio Web, un documento con toda esta descripción. Este documento está construido en WSDL. En el registro UDDI correspondiente al servicio, se hace referencia a la URL donde está este documento WDSL. Ese documento WSDL tendrá toda la información necesaria para acceder al XML Servicio Web.

El documento WSDL describe como los argumentos de invocación al servicio incluyendo tipo de dato y nombre, para la invocación HTTP GET, HTTP POST y SOAP.

Si bien el proveedor del servicio puede hacer cambios en la lógica dentro de los métodos, mientras no modifique la invocación, el documento WSDL no cambiará. Se podría, por ejemplo, cambiar las bases de datos a las cuales se conecta el servicio, y el cliente lo invoca no se enteraría.

3.2.7.12.3. El consumidor y el retorno del XML Servicio Web

Evidentemente, XML Servicio Web se basa en un modelo donde existe un proveedor (la aplicación que expone el XML Servicio Web) y un consumidor (la aplicación que usa el XML Servicio Web).

Supongamos que un usuario de una aplicación usa el XML Servicio Web. Cuando se dice que el servicio retorna un documento XML, el usuario no debería necesariamente, ver ese documento. Es más: lo deseable, para obtener la abstracción deseada es que no lo vea.

La aplicación que recibe la respuesta del XML Servicio Web, debe estar preparada para “consumir” ese XML, y transferir la información que está en él a la aplicación de usuario, de una manera amigable.

O sea: una aplicación Web actúa como consumidor, invoca al XML Servicio Web, recibe el documento XML que retorna, y lo muestra de manera apropiada al usuario.

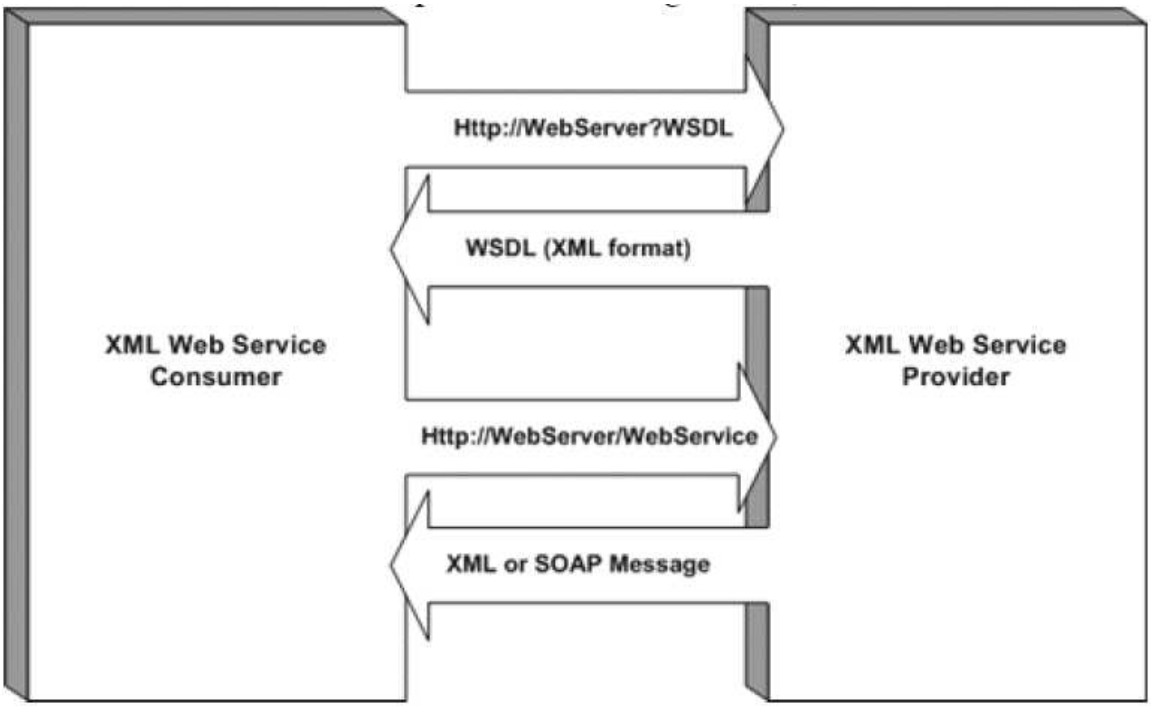

Figura 3.9. Ciclo de un XML Servicio Web [MOL 2004]

El ciclo de uso de un XML Servicio Web sería entonces:

1. El usuario que necesita el servlcio, lo busca en el registro UDDI (o accederlo directamente si se tiene la URL donde está).
2. Si está, requiere el documento WSDL.
3. El proveedor envía el documento WSDL en formato XML.
4. El proveedor hace un requerimiento del XML SERVICIO WEB que figura en el WDSL, pasando los argumentos según la forma enunciada en el WSDL.
5. El proveedor procesa el requerimiento y retorna el resultado en un documento XML o en un mensaje SOAP.

3.2.8. SOAP

Para introducirnos en el concepto de SOAP, pensemos en esencia que es lo que se hace con un Servicio Web. El Servicio Web es invocado, a través de parámetros de entrada y retorna un resultado, como parámetro de salida.

Bueno, pensemos en esa comunicación como si fuera un sobre con información que es intercambiada. SOAP define, por un lado, ese “sobre” que envuelve la información.

Especifica una serie de reglas para representar la información que viaja. Pero tiene, además, una función muy importante: definir una convención la invocación a procedimientos remotos y su respuesta. Resumiendo: SOAP define el sobre, las reglas y la convención de invocación remota, esta última, opcional.

El éxito de SOAP es que está basado en texto (usando XML) lo que lo hace independiente de distintas plataformas (y vendedores). De esta forma al invocar un método, no es necesario saber en qué lenguaje está desarrollado, si es remoto o no, etc.

SOAP no incluye, por sí mismo:

- Garbage collection distribuido
- Manejo de objetos por referencia (requiere garbage collection)
- Activación (requiere manejo de objeto por referencia)

SOAP es lo que se llama “lightweight protocol”. Tiene capacidades fundamentales como:

- Poder enviar y recibir paquetes HTTP o de otros protocolos
- Poder procesar XML.

Lo interesante de estas tecnologías que utilizan XML y SOAP es que pueden integrarse en aplicaciones ya existentes sin demasiada complejidad. Productos que usan SOAP, pueden usar servidores HTTP existentes y procesadores XML que tenga el cliente.

Si en cambio, se utiliza CORBA, DCOM o Java RMI se debe instalar el ambiente apropiado, configurar el sistema para adaptarlo a la nueva infraestructura distribuida. También puede exigir reconfigurar firewalls. Por eso a esta tecnología se lo asocia con “heavyweights systems”.

SOAP permite serializar la invocación de métodos remotos, transformando la llamada al método en la forma adecuada para su transmisión sobre la red, sobre una capa de transporte que lleve esos datos inherentes al método entre los sistemas remotos.

3.2.8.1. SOAP y su relación con otros protocolos

Si bien SOAP puede usar otros protocolos de transporte, el uso de HTTP permite ampliar el rango de los puntos de acceso, al pasar fácilmente a través de firewalls (otros protocolos tienen los puertos deshabilitados por seguridad).

SOAP puede integrar sistemas que hasta ahora no podían comunicarse. Y esto es gracias a XML. Todos los mensajes SOAP son codificados usando XML. SOAP define dos namespaces:

- El del envelope, http://schemas.xmlsoap.org/soap/envelope
- Para la serialización, http://schemas.xmlsoap.org/soap/encoding

SOAP fue diseñado para transformar los parámetros de la invocación al método desde su forma nativa (binaria) a XML, donde en el puerto de destino el procesador SOAP toma esa información XML y lo lleva a un estado propio del destino para que pueda procesarse.

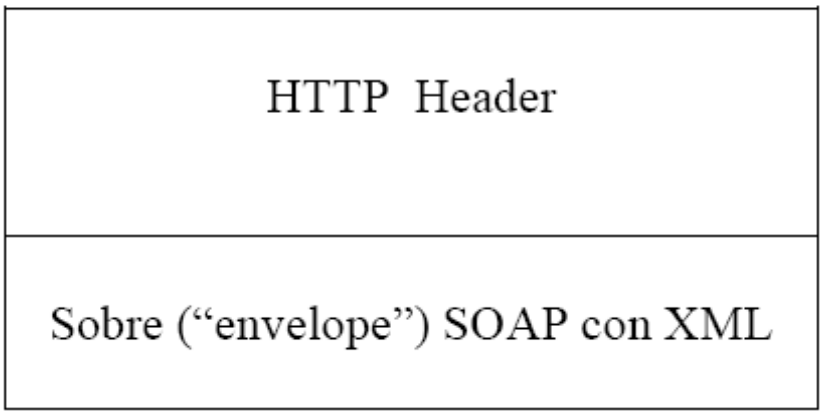

Figura 3.10. Paquete http [MOL 2004]

En el header HTTP se indica el destino, en qué está el contenido del paquete (texto basado en XML). Otra ventaja de SOAP, es que permite serializar la invocación de métodos remotos usando un protocolo de serialización disponible y listo para ser usado, sobre la capa de transporte que requiera la computación ubicua.

SOAP permite implementar un mecanismo de serialización y el transporte necesario de los datos de intercambio en la invocación remota de métodos.

SOAP no define por sí mismo ninguna semántica de aplicación (por ejemplo, un modelo de programación) ni cubre aspectos de activación/desactivación, garbage collection, o seguridad. Es decir: SOAP no es una arquitectura entera distribuida.

3.2.8.2. El modelo de intercambio de mensajes de SOAP

Fundamentalmente los mensajes SOAP son de transmisión de una sola vía (one way). Pero pueden combinarse para ofrecer un modelo request/response. Por ejemplo, se puede implementar que los mensajes de respuesta SOAP sean librados como respuestas HTTP.

3.2.8.3. Qué pasa cuando llega un mensaje SOAP

Los mensajes que llegan, no necesariamente son para ese nodo. El mensaje puede seguir un “message path”, y tocar nodos intermedios.

Cuando llega un mensaje SOAP, la aplicación que lo recibe debe realizar los siguientes pasos:

- Identificar todas las partes del mensaje
- Verificar que todas las partes obligatorias son soportadas por la aplicación y procesarla. Si no es así, descartar el mensaje.
- Si no es el lugar de destino, remover las partes que correspondan antes de forwardear el mensaje.

El procesador SOAP del mensaje debe contar con la información necesaria para saber qué modelo de intercambio se usa, si se usan mecanismos de RPC, cómo se representan los datos, etc.

3.2.8.4. Estructura SOAP

SOAP como protocolo, consiste de tres partes:

- El **envelope**, (que hay en el mensaje, a quién va dirigido, si es opcional o mandatorio)
- **Reglas de codificación**, que definen el mecanismo de serialización para el intercambio de tipos de datos definidos por la aplicación
- **Una representación RPC** que define la convención sobre invocación remota de métodos y su respuesta

En un mensaje SOAP se distingue:

- Un envelope, obligatorio
- Un header, opcional
- Un body, obligatorio

- **El "envelope"**

El "envelope" SOAP envuelve la información a transmitir, sin especificar una dirección. Está formado por grupos de notas e información separados por tabs. Es el elemento de mayor nivel. Puede contener declaraciones de namespaces y atributos. Si están atributos deben estar calificados por namespaces. Puede contener subelementos.

- **El "header"**

El header es opcional. Tiene información para manejo y control, como números de cuenta o identificación de cliente. Puede tener también información sobre el algoritmo usado para encriptar el body o la

clave pública necesaria para leer el texto encriptado o un identificador de transacción o algún mecanismo para prevención de deadlock.

Si está presente, debe ser el primer elemento hijo del elemento envelope. Puede usarse el atributo encodingStyle, y también los atributos mustUnderstand y SOAP actor.

- **El "body"**

  En el Body se registra la información a transportar. Debe ser un elemento hijo del envelope, y debe seguir al header si este está presente.

  Provee un mecanismo para intercambiar información con el receptor último del mensaje. Normalmente incluye el marshalling para llamadas RPC y reportes de error.

  Las reglas de codificación del body incluyen:

  - La entrada debe ser identificada por un nombre de elemento totalmente calificado, por un namespace URI y nombre local. Los elementos hijos pueden ser calificados por namespace.
  - Puede usarse el atributo encodingStyle para indicar la serialización.

3.2.9. WDSL (Servicio Web Description Language)

3.2.9.1. Generalidades sobre WDSL

Cuando un proceso cliente y un proceso servidor quieren interactuar, los que desarrollan el software deben acordar cuál es el nombre de la operación que invocará la aplicación cliente, como la invocará (parámetros de entrada, tipos) y qué responderá la aplicación servidor.

En COM y CORBA, estas interfases y las operaciones que ellas describen están construidas en IDL (Interface Definition Language). En Servicio Web se necesita algo similar. Y para esta tecnología se usa WDSL.

Para explicar porqué se prefiere usar WSDL en lugar de COM, por ejemplo, valga aclarar que la IDL que se usa en COM tiene una sintaxis derivada de C. En cambio, WSDL se define usando XML. Y ahí está la razón: el uso de un protocolo estándar en Internet que facilita la interoperabilidad.

WSDL no es simple. Para hacer una interfase WSDL se debe tener muy buen conocimiento de XML. La ventaja es que “entendido” por prácticamente cualquier plataforma.

3.2.9.2. WSDL y SOAP

WSDL extiende los beneficios de SOAP, al proveer una forma de que los usuarios y proveedores de Servicios Web se puedan comunicar y trabajar más allá de plataformas y lenguajes.

SOAP no necesita un formato que lo describa. En cambio WSDL es imprescindible como lenguaje de descripción, pues sino no sabremos cómo invocar el Servicio Web, su nombre, sus argumentos.

Si bien esta comunicación podría establecerse estáticamente, e involucrar al usuario (averiguando y luego invocando) se pierde la característica de dinamismo del modelo de Servicios Web. WSDL puede hacer bindings a otros protocolos, no sólo a SOAP. Pero para obtener la interoperabilidad deseada, se recomienda SOAP sobre HTTP.

3.2.10. UDDI

Supongamos que un desarrollador necesita encontrar la interfase WSDL para implementar un servicio. No sólo debe saber si ya existe, sino cómo es el intercambio de forma compatible.

UDDI (Universal Description, Discovery and Integration) es una especificación para registros distribuidos de información sobre Servicios Web. Es tanto una especificación para registrar como para buscar.

Para mostrar los servicios que ofrece una empresa, puede crear una registración UDDI, que es nada menos que un documento XML (cuyo formato especifica un esquema UDDI).

Cuando se crea, esta registración es almacenada en una base que se llama ***UDDI Business Registry***. Y una copia se mantiene en cada site de

operadores UDDI, replicada. Estos sites están comunicados para mantener la consistencia.

El UDDI Business Registry es usado tanto por programas como por programadores para encontrar información acerca de servicios y para saber cómo invocarlos y trabajar con ellos. También es usado para publicar lo propio para que pueda ser consumido por otro.

El UDDI es en sí mismo un Servicio Web, que provee información para poder utilizar otros Servicios Web. Los desarrolladores que quieran escribir software para acceder o modificar la información en la registry, deben usar una API definida por UDDI.

El esquema XML UDDI sirve para definir una estructura común para todos los documentos de registración. La información que se registra en el UDDI tiene una orientación a la lógica de negocio. Está organizada en:

- ***White pages***: información para contactar a la compañía, tal como nombre, descripción de la compañía, identificadores tipo CUIT, etc.
- ***Yellow pages***: información sobre tipo de negocio, tal como índice de productos o servicios que comercializa.
- ***Green pages***: información técnica sobre los servicios que expone, tal como reglas de negocio, descripción de servicios, invocación a aplicaciones, binding de datos.

3.2.11. Servicios Web en .NET

3.2.11.1. El modelo .NET

.NET es un conjunto de tecnologías de Microsoft. Es más una plataforma que un producto. Para simplificar podemos verlo también como un framework para aplicaciones.

Sobre esta tecnología pueden construirse aplicaciones y servicios que funcionan independientemente de la plataforma. Permite la comunicación entre diversos sistemas y aplicaciones, integrando distintos dispositivos cliente.

El objetivo de Microsoft con la plataforma .NET fue establecer una nueva generación de sistemas, integrando la funcionalidad de sistemas sobre Internet. Ha sido pensada como una plataforma para sistemas altamente distribuidos e interoperables, tratando de simplificar el acceso a la Web, a través de cualquier dispositivo y usando cualquier plataforma.

3.2.11.2. XML Servicios Web en Microsoft

Un servicio XML WS es una forma simple para que los objetos en un servidor acepten requerimientos desde clientes usando HTTP/XML.

Para escribir XML WS, se escribe un objeto .NET como si fuera a accederse localmente y se marca para indicar que puede accederse desde la Web. Luego ASP.NET se encarga de crear la infraestructura necesaria que acepta los requerimientos HTTP que llegan y lo relaciona con el objeto.

Del lado del cliente, .NET provee clases proxy para poder acceder fácilmente a cualquier servidor mediante HTTP. En tiempo de programación, el desarrollador crea un código objeto proxy desde la descripción de un WS XML Hacia y desde el servidor

3.2.12. WSDL en .NET

ASP.NET puede generar una descripción examinando el metadata de un assembly. Un assembly es una colección de uno o más EXE o DLL que contienen el código de la aplicación y recursos.

También tiene un "manifiesto" o declaración (metadata) donde se describen el código y los recursos que están "dentro" del assembly. Esa descripción se almacena en un archivo XML con el vocabulario que usa WSDL.

WSDL se puede obtener desde ASP.NET, al requerir el archivo .ASMX con ?wsdl agregado a la URL.

3.2.13. SOAP en .NET

ASP.NET acepta tres formas de requerimientos que ingresan: HTTP GET, HTTP POST y SOAP. Los dos primeros se mantienen por compatibilidad.

Cuando el cliente invoca un método en el proxy, este invoca al método *Invoke* que crea un mensaje SOAP que contiene el nombre del método y los parámetros del método a invocar XML WS y lo envía al server sobre HTTP.

Cuando vuelve el paquete SOAP, la clase base hace el parse, para tomar el valor de retorno y se lo pasa al proxy.

3.3. Selección de la Tecnología para la Solución del Problema

Se presentará una serie de ventajas y desventajas de las tecnologías anteriormente mencionadas para terminar en una matriz comparativa mediante la cual se seleccionará la alternativa de solución adecuada:

3.3.1. Comparación de plataformas

Criterios para la elección de la plataforma [MSDAS 2006]:

- Capacidades de desarrollo.
- Soporte multiplataforma.
- Amplio uso y aceptación.
- Estabilidad.
- Disponibilidad de herramientas.
- Aplicabilidad a técnicas actuales de diseño (XML, Servicios Web).
- Rendimiento.
- Facilidad de uso.
- Tiempo de desarrollo.

3.3.1.1. Ventajas de CORBA frente a las demás tecnologías:

- Abstrae a la aplicación de todo lo referente a las comunicaciones. El programa ni siquiera sabe donde esta el objeto al que llama.
- Se pueden reutilizar programas anteriores simplemente añadiendo algo de código.

- Tiene todas las ventajas de la programación orientada a objetos.
- Se puede programas en varios lenguajes. Puede crearse un programa con varios lenguajes distintos, ya que cada una de las partes solo verá el interfaz CORBA de la otra.
- Las simplificaciones en la programación deberían llevar a una programación con menos errores.
- Abstrae el Sistema Operativo.
- Es un estándar para todas las plataformas y abierto.
- Es escalable.

3.3.1.2. Desventajas de CORBA frente a las demás tecnologías:

- Falta de propietarios.
- No logró implicar a los mayores vendedores (Microsoft).
- Protocolo de transporte opaco.
- Dependencia de comunicaciones punto a punto.
- Mucho esfuerzo en desarrollo.
- Mucho desarrollo pendiente.
- Más capas de software, más carga.
- Usa RPC en su nivel más bajo, con los problemas que esto acarrea.
- CORBA es muy grande: un fallo en CORBA podría ser difícil de detectar.
- Aunque se necesite añadir poco código, se necesita reescribir las aplicaciones.

3.3.1.3. Ventajas de RPC frente a las demás tecnologías:

- Se ocultan detalles de la comunicación
- No hay diferencias entre una llamada local y una remota
- Tiene una alta intuitividad.

3.3.1.4. Desventajas de RPC frente a las demás tecnologías:

- Uso de múltiples componentes
- No es posible el Balanceo de Carga y Tolerancia a fallos.
- Con RPC es muy difícil detectar que servidores están con mucha carga de trabajo y derivar la llamada RPC a otro servidor menos ocupado
- RPC no puede manejar los picos de carga de trabajo que puede tener un servidor si tiene llamadas RPC de muchos clientes.

3.3.1.5. Ventajas de DCOM frente a las demás tecnologías:

- Hay muchos libros, herramientas y desarrolladores.
- Existe una buena integración con Visual Basic y JAVA.
- Microsoft depende de su funcionamiento.

3.3.1.6. Desventajas de DCOM frente a las demás tecnologías:

- Arquitectura compleja
- Escalabilidad difícil de implementar
- No sigue estándares abiertos

3.3.1.7. Ventajas de RMI frente a las demás tecnologías:

- Portable a través de plataformas con soporte JAVA.
- Bajo costo al convertir sistema existente.
- Soporta paso de objetos por referencia y/o valor.
- Es REALMENTE fácil de implementar si ya se conoce JAVA.

3.3.1.8. Desventajas de RMI frente a las demás tecnologías:

- A veces, no es tan intuitivo.
- No soportado por otros lenguajes
- Disminuye el rendimiento con el crecimiento del sistema

3.3.1.9. Ventajas de .NET Remoting frente a las demás tecnologías:

- Arquitectura para distribuir objetos en forma sencilla.
- Muy sencilla de utilizar para el desarrollador (los detalles pasan inadvertidos).
- Marco eficaz, extensible e independiente del lenguaje que permite desarrollar sistemas distribuidos sólidos y escalables.
- Servicios remotos integrados a la perfección con los servicios Web y permiten exponer objetos .NET para tener acceso en varias plataformas.

3.3.1.10. Ventajas de los Servicios Web frente a las demás tecnologías:

- Aportan interoperabilidad entre aplicaciones de software independientemente de sus propiedades o de las plataformas sobre las que se instalen.

- Los servicios Web fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento.
- Al apoyarse en HTTP, los servicios Web pueden aprovecharse de los sistemas de seguridad firewall sin necesidad de cambiar las reglas de filtrado.
- Permiten que servicios y software de diferentes compañías ubicadas en diferentes lugares geográficos puedan ser combinados fácilmente para proveer servicios integrados.
- Permiten la interoperabilidad entre plataformas de distintos fabricantes por medio de protocolos estándar y abiertos. Las especificaciones son gestionadas por una organización abierta, la W3C, por tanto no hay secretismos por intereses particulares de fabricantes concretos y se garantiza la plena interoperabilidad entre aplicaciones.

3.3.1.11. Desventajas de los Servicios Web frente a las demás tecnologías:

- No son tan desarrollados para realizar transacciones, comparado a otros sistemas como CORBA (Common Object Request Broker Architecture).

- Su rendimiento es bajo comparado con otros sistemas como CORBA, DCOM o RMI, especialmente por el uso de protocolos y estándares basados en texto.

3.3.2. Comparación entre las tecnologías .NET y JAVA para el desarrollo de los Servicios Web [SK 2003]

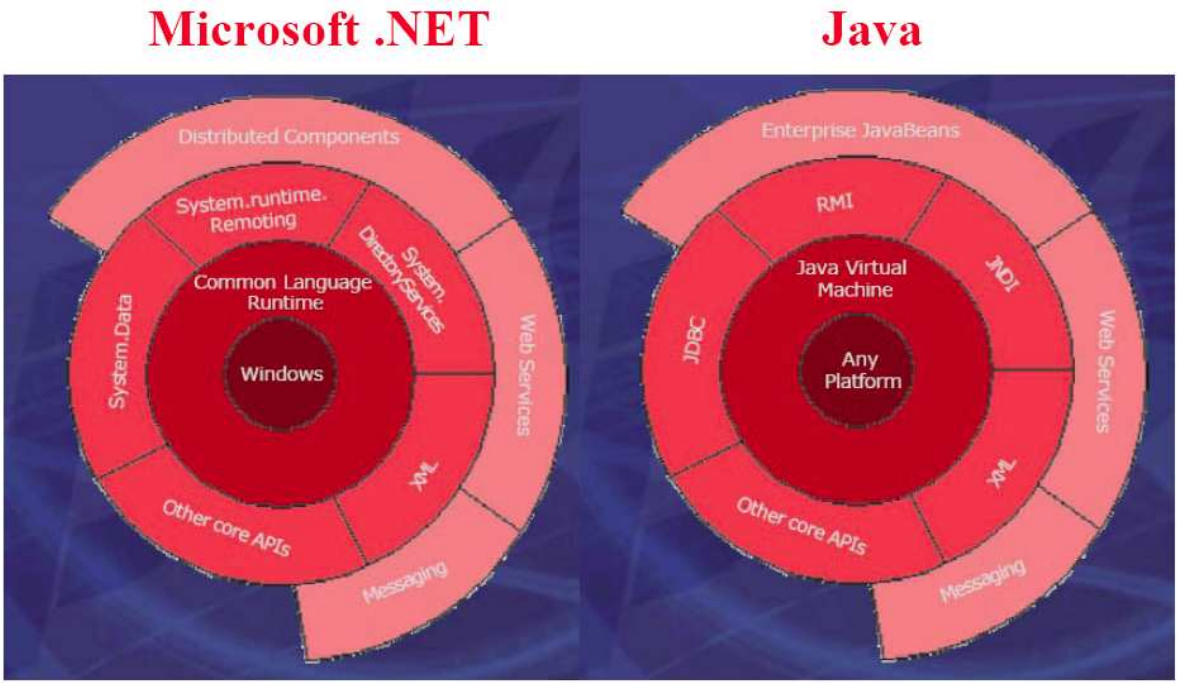

Figura 3.11. Tecnologías .NET vs. Java [SK 2003]

3.3.2.1. Filosofías de diseño:

3.3.2.1.1. Microsoft .NET

- Entorno de trabajo centralizado (Windows) e independiente del lenguaje de programación.

3.3.2.1.2. Java 2 Platform Enterprise Edition (J2EE)

- Lenguaje de programación centralizado e independiente del entorno de trabajo.

3.3.2.2. Características comunes:

- Entorno de desarrollo completo y robusto.
- Orientado al desarrollo de entornos distribuidos escalables.
- Servicios Web.

- Modelos de componentes distribuidos.
- Data services.

3.3.2.3. Ventajas de .NET

- Entornos de desarrollo.
- Servicios Web.
  - WebService, ASP.
- Servicios poco acoplados.
  - SOAP, WSDL, UDDI.
- Servicios de datos/directorios.
  - ADO.NET, System.DirectoryServices.
- Servicios distribuidos transaccionales.
  - COM+.
- Objetos distribuidos.
  - .NET Remoting.

3.3.2.4. Ventajas de Java

- Servicios Web.
  - Servlets, Java Server Pages (JSP).
- Servicios poco acoplados.
  - XML APIs, Java Messaging Service (JMS).
- Servicios de datos/directorios.
  - Java Database Connectivity (JDBC).
  - Java Naming y Directory Interface (JNDI).
- Servicios distribuidos transaccionales.

- Enterprise JavaBeans (EJB) Components.
- Java Transaction Service (JTS/XA).

- Objetos distribuidos.
  - RMI, CORBA.

Característica	J2EE	.NET
Intérprete	JRE	CLR
Página Web dinámicas	JSP	ASP .NET
Componentes Middle-Tier	EJB	Componentes manejados por .NET
Acceso a BD	JDBC, SQL/J	ADO .NET
SOAP, WSDL, UDDI	Si	Si
Otras características como load balancing	Si	Si

Tabla 3.3. Comparativo entre J2EE y .NET

3.3.2.5. Ventajas y desventajas que ofrecen estos modelos

Con respecto a ambas:

- Exigen capacitación de los desarrolladores. En el caso de J2EE, en Java, y en el caso de .NET, en Orientación a Objetos.
- En cualquiera de ellas se pueden construir Servicios Web.
- Ambas tienen un bajo costo de sistema. En el caso de J2EE se puede usar jBoss7 o Linux, y en el caso de .NET, Windows/Win32.
- Ambas ofrecen, teóricamente, escalabilidad y extensibilidad.

3.3.2.5.1. *.NET supera a J2EE en:*

- .NET comenzó con Servicios Web antes que J2EE.
- Tiene más experiencia en shared context.
- Tiene un modelo de programación más simple.
- Mantiene una neutralidad de lenguaje cuando se desarrollan nuevas aplicaciones de Ebusiness, mientras que J2EE trata a los otros lenguajes como aplicaciones separadas.

3.3.2.5.1. *J2EE supera a .NET en:*

- Es una plataforma probada, con nuevas API's para Servicios Web.
- El código ya desarrollado en J2EE se puede llevar fácilmente a J2EE Servicios Web, con poca reescritura, lo que no ocurre en la portabilidad de Windows
- DNA y .NET
- .NET Servicios Web no es interoperable con standards de la industria. El framework BizTalk tiene extensiones SOAP que son propietarias y no soportan ebXML.

3.3.3. Matriz de comparación de tecnologías para elección de la alternativa de solución.

	Servicios Web	RMI	CORBA	.NET Remoting
Protocolos	SOAP http [COMPVII 2008]	JRMI [COMPII 2008]	IIOP, GIOP [COMPI 2008]	TCP, HTTP, SMTP [COMPV 2008]
Interoperabilidad	Soporta interoperabilidad entre entornos heterogéneos. [COMPVI 2008]	No soportado por otros lenguajes. [COMPVIII 2008]	Provee soporte a través de varias plataformas. [COMPIX 2008]	Necesita ambientes homogéneos. [COMPIII 2008]
Escalabilidad	Ofrece escalabilidad lo que nos permite interceptar los mensajes SOAP durante las etapas de serialización y deserialización. [COMPVII 2008]	Disminuye el rendimiento con el crecimiento del sistema. [COMPII 2008]	Es escalable en sistemas grandes a enormes. [COMPI 2008]	Muy escalable por lo que nos permite personalizar los diferentes componentes del Framework NET Remoting. [COMPIII 2008]
Fácil Programación	Fácil de crear y desplegar. [COMPVI 2008]	Desarrollo rápido y fácil de objetos distribuidos. [COMPIV 2008]	Complejo de programar. [COMPIV 2008]	Complejo de programar. [COMPIII 2008]

Tabla 3.4. Comparativo entre tecnologías distribuidas.

La matriz 3.4 presenta un resumen de la comparación entre las principales tecnologías de objetos distribuidos y en la cual se observa que con los Servicios Web se tiene mayor facilidad de implementación y nos garantiza un funcionamiento de acuerdo a los requerimientos demandados por el problema presentado (Escalabilidad, Interoperabilidad, etc.) por lo que es la alternativa que se escogió para la presentación de la solución.

Cabe señalar que se utiliza SOA con Servicios Web y no SOA con otras tecnologías (REST [WIL 2008], CORBA, etc.), por lo siguiente:

- No hay herramientas para implementar REST rigurosamente; sí en cambio las hay para SOAP RPC. [REY 2008]
- Los web services no son parte obligatoria de SOA, pero son una implementación adecuada. [REY 2008]
- "SOA ha surgido como la mejor manera de afrontar el desafío de hacer más con menos recursos. Promete hacer la re-utilización y la integración mucho más fáciles, ayudando a reducir el tiempo de desarrollo y aumentando la agilidad organizacional. No sorprendentemente, el 80% de las organizaciones de IT están implementando aplicaciones usando SOA con web services subyacentes. SOA proporciona mayor flexibilidad para afrontar los cambios tanto en el ambiente de negocios como en la infraestructura tecnológica" [SCH 2005]
- "Comprender el rol y el significado de SOA, más allá del hype simplista, es imperativo para cualquier arquitecto de software empresarial. ... Hacia 2008, SOA y Web Services serán implementados juntos en más del 75% de los proyectos que utilicen SOA y Web Services (probabilidad 0.7)" [SCH 2005]

3.4. Aplicativos existentes en la actualidad

Actualmente existe un aplicativo utilizado por la SEPS y las Entidades vinculadas al Sistema EPS que realiza las transacciones electrónicas de datos, es el denominado SITEDS (Sistema Integrado de Transacciones Electrónicas de Datos en Salud), un aplicativo realizado siguiendo los lineamientos de EDI (Intercambio electrónico de datos) y el estándar ASC X12 [X12 2008].

El SITEDS funciona bajo el Sistema TEN (Transaction Environment Network) que es un conmutador de mensajes bajo UNIX y se divide en 3 componentes

- SITEDS Cliente (Aplicativo Visual Basic para el usuario final).
- SITEDS EPS (donde se encuentra la Base de datos).
- SITEDS SEPS (recibe y envía las transacciones de datos).

El detalle del funcionamiento del SITEDS se presenta en el anexo 01 adjunto a la presente investigación.

3.5. Caso de Estudio

El caso de estudio donde aplicamos la solución propuesta al problema de la presente investigación fue desarrollado en la SEPS (Superintendencia de Entidades Prestadoras de Salud), se tomó el módulo SITEDS Cliente como aplicativo para el usuario final y para el paso de transacciones de consulta se implementaron Servicios Web los cuales se encuentran alojados en un servidor de la EPS Mapfre Perú S.A.

Para empezar el desarrollo del caso de estudio primero se creó los stores procedures correspondientes a la lógica de negocios necesaria para que las Entidades Vinculadas al Sistema de EPS puedan realizar consultas de sus afiliados de la base de datos de Mapfre EPS.

Seguidamente se empezaron a crear los Servicios Web consumidores de la lógica del negocio, implementados con los métodos de seguridad y autenticación correspondientes. Una vez desarrollados estos dos puntos se pidió a Mapfre que pasara a producción estos stores y Servicios Web.

Para el usuario final se tuvo que modificar una pequeña parte del aplicativo SITEDS Cliente para que ya no utilice al TEN Server como componente transaccional sino que se sirva de los Servicios Web como el realizador de las transacciones de consulta. La solución propuesta en el caso de estudio presenta varios beneficios como por ejemplo:

- Tiempo de desarrollo e implementación relativamente corto (aproximadamente un mes)

- Bajos costos de desarrollo
- Altamente Escalable
- Flexibilidad
- Interoperabilidad

Estos beneficios son muy significativos a comparación de lo que se venía manejando en la SEPS con el SITEDS ya que abre una nueva gama de posibilidades de implementación inherentes a los Servicios Web (el SITEDS utiliza tecnologías que dentro de poco serán caducas, si es que ya no lo son).

A continuación se presentan pantallas de la implementación del caso de estudio en la SEPS:

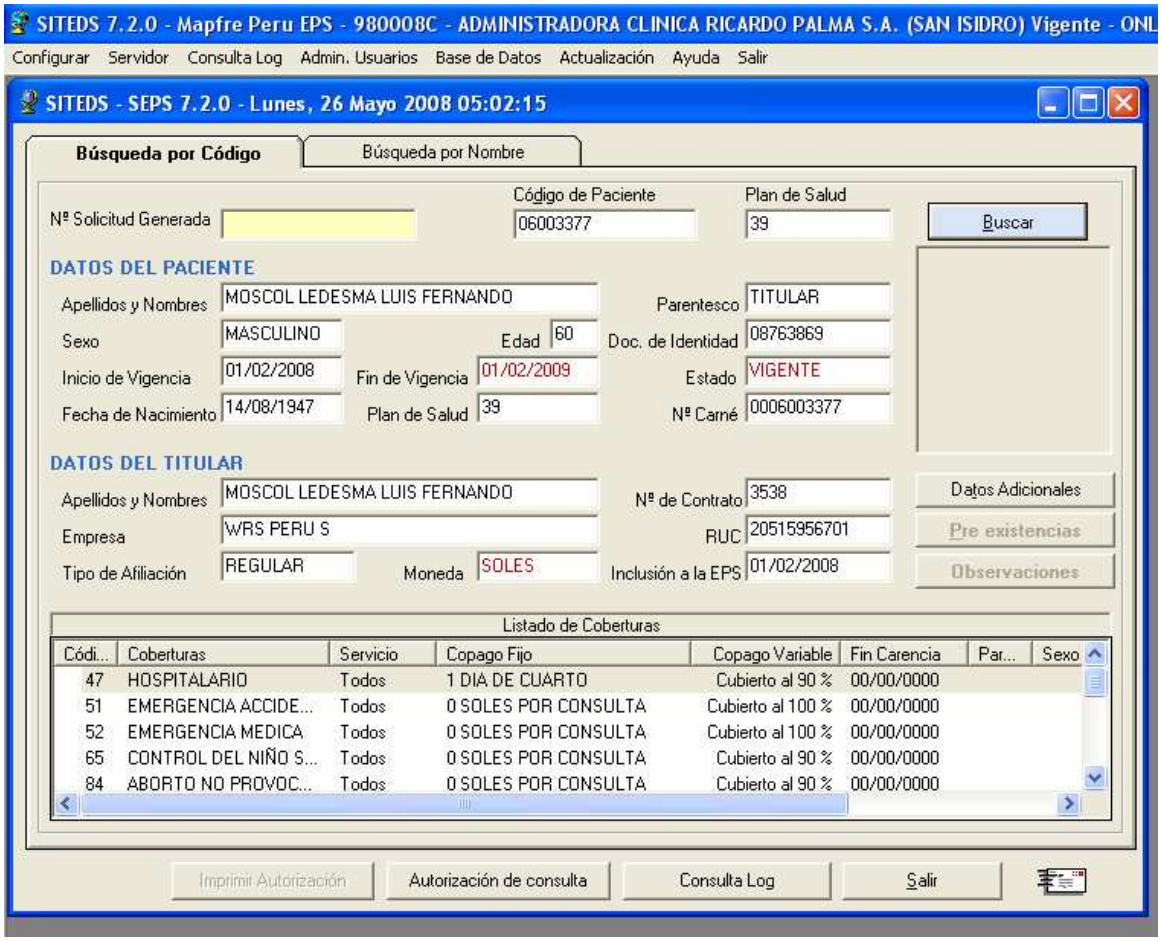

Figura 3.12. Pantalla principal del SITEDS Cliente

La figura 3.12 muestra la pantalla principal del SITEDS Cliente, en la cual se ha consultado por el afiliado **Moscol Ledesma, Luis Fernando**; esta consulta puede realizarse desde cualquier PC donde esté instalado el SITEDS Cliente, internamente llama a los Servicios Web colgados en Mapfre EPS para devolverle la trama de respuesta y mostrársela al usuario final.

CAPITULO IV

SOLUCION AL PROBLEMA

Independientemente de cómo se considere el diseño de software basado en la arquitectura SOA es muy potente [JD 2006]. Los beneficios de la arquitectura SOA resultan muy reales y se están extendiendo desde empresas individuales a sectores industriales enteros [JD 2006].

La solución planteada es implementar una solución informática basada en la Arquitectura orientada a Servicios empleando los Servicios Web, en reemplazo a los componentes servidores del SITEDS, puesto que los Servicios Web son componentes que permiten crear sistemas distribuidos abiertos, lo que hace posible que distintas aplicaciones, de diferentes orígenes, comunicarse entre ellos sin necesidad de escribir programas costosos, esto porque la comunicación se hace con XML.

Los Servicios Web no están ligados a ningún Sistema Operativo o Lenguaje de Programación. Por ejemplo, un programa escrito en Java puede conversar con otro escrito en .Net; aplicaciones Windows puede conversar con aplicaciones Unix. Por otra parte los Servicios Web no necesitan usar browsers (Internet Explorer) ni el lenguaje de especificación HTML.

4.1 Solución al Problema

Los componentes de servidor (EPS y SEPS) del SITEDS, serán reemplazados cada uno por un Servicio Web. El cambio en estos componentes es casi total, puesto que con esta solución las transacciones del negocio serán expuestas por los Servicios Web.

- **Servicio Web SEPS**
  - Es el encargado administrar y enrutar de mensajes, se alojara en un servidor de aplicaciones de la SEPS, que se encuentra alojado en el nodo central de dicha institución.
  - Es el responsable de la Autentificación de los usuarios que deseen consumir los servicios, estos pueden ser las entidades vinculadas con desarrollo propio o el componente cliente SITEDS.
  - Responsable del cifrado de la información a retornar.
  - El descubrimiento del servicio es detallado en el Anexo 03.
  - Interactúa con la Base de datos de la SEPS , para el registro de las Transacciones realizadas

- **Servicio Web EPS**
  - Es el encargado de recibir las transacciones y responder a estas, luego de interactuar con los sistemas informáticos propios de la EPS, será alojado en un servidor de aplicaciones de la EPS.
  - Es el responsable de la generación de la Trama de respuesta que se obtiene luego de procesar la consulta solicitada.

- Interactúa con la Base de datos de la EPS.
- El descubrimiento del servicio es detallado en el Anexo 03.

Para poder aminorar el impacto de la implantación de la nueva solución, es necesario seguir empleando el formato de la trama de exposición de datos de los asegurados, específicamente la data que se muestra en el componente cliente, por los siguientes motivos:

- *El mantenimiento al Componente Cliente del SITEDS debe ser mínimo*, esto se conseguirá con solo agregar las rutinas necesarias para consumir el Servicio Web, empleando como componentes para la establecer conectividad con un Assembly [NET 2006] desarrollado en Visual Basic .Net, con cual es sistema realizara una interoperabilidad. Dichas librería serán utilizadas en el componente cliente del SITEDS, puesto que dicho componente esta desarrollado en Visual Basic 6.

- *Existen Clínicas con desarrollo propio que tiene implementado los algoritmos de traducción de data de la trama EPS*, muchas clínicas que cuentan con una buena infraestructura tecnológica, tienen integrado a sus sistemas internos las transacciones del SITEDS, desde el modo de invocarlo hasta el proceso de interpretación de la trama que se obtiene como resultado de consulta, por lo que un gran cambio en el resultado que se obtiene en la consulta información seria muy costoso para esas entidades de salud.

Por lo expuesto, se conservara el formato de la Trama EPS, obteniendo como respuesta a la consulta de los métodos Web, una cadena de datos con la estructura EPS.

El Servicio Web SEPS, será el responsable de enrutar los mensajes provenientes de las Entidades Vinculadas, este administrara y resolverá a que Servicio Web enviar el mensaje, basándose primero en la autenticación y autentificación del usuario, luego en los parámetros de input de los métodos Web.

En la Figura 4.1 se puede observar la arquitectura propuesta para el SITEDS.

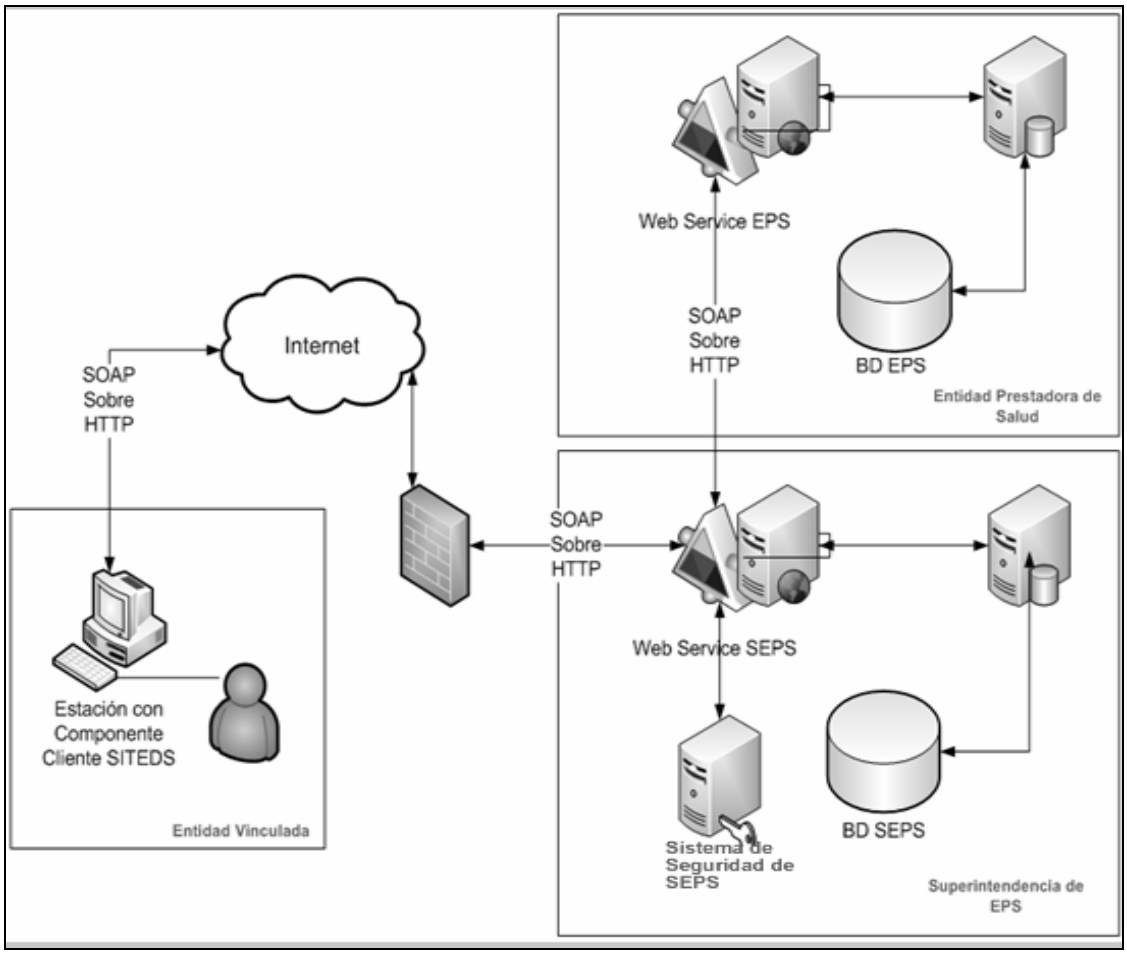

Figura 4.1. Arquitectura del SITEDS basado en Servicio Web

La solución propuesta se basa la definición de 5 operaciones que responden a las funcionalidades críticas de los componentes servidores del SITEDS. A continuación se describe brevemente cada una de ellas.

Operación 1	**GetConsultaxnombre**
Descripción	Entrega como información el listado de asegurados coincidentes con el criterio de consulta solicitada. Por Ejemplo, al ser consultado pacientes por el apellido paterno Orbezo este entrega como resultado todos los asegurados que coincidan con el apellido paterno Orbezo.

Operación 2	**GetConsultaxCodigo**
Descripción	Entrega como resultado los datos generales del asegurado, así como también las coberturas del asegurado, de acuerdo a su contrato de plan de salud, en la Entidad Vinculada donde desea atenderse. Por ejemplo , el asegurado Juan Pérez ,código 01452896 , de sexo masculino ,de número de documento de identidad 40178380 ,etc.

Operación 3	**GetCodigoAutorizacion**
Descripción	Entrega como resultado un código único de autorización la atención del asegurado para la cobertura seleccionada. Por Ejemplo, código 012546 para la autorización de la atención del asegurado Juan Pérez se atiende por la cobertura de Emergencia Medica de código 52.

Operación 4	**GetFotoAsegurado**
Descripción	Entrega como resultado en un mensaje de formato Byte, donde se agrega la foto del asegurado en formato binario.

Operación 5	GetDatosOtros
Descripción	Entrega como resultado los datos adicionales del asegurado o las observaciones que existen sobre su atención.

Los Servicios Web EPS y SEPS presentan las mismas funciones, pero no la misma funcionalidad, puesto que el Servicio Web SEPS es el enrutador de las transacciones.

Las Tablas 4.1, 4.2, 4.3, 4.4 .4.5 como parámetros de entrada, presentan una lista de los parámetros obligatorios para cada una de las funciones.

Para todas las tablas se presentan el nombre con que debe ir el parámetro en la llamada, la descripción que indica su uso, el tipo de parámetro y finalmente los valores posibles que puede tener cada parámetro.

Nombre	Descripción	Tipo	Valores
Entidad_Codigo	Código de la EPS a quien se realiza la consulta .Ejemplo : 060027ª	String	Código de Registro de la EPS proporcionada por la SEPS.
EVinculada_Codigo	Código de Registro de la Entidad Vinculada. Ejemplo 980005C	String	Código de Registro de la Entidad Vinculada
EVinculada_Ruc	RUC de la Entidad Vinculada. Ejemplo : 20100162742	String	RUC de la Entidad Vinculada
PaternoAsegurado	Apellido Paterno que se desea consultar. Ejemplo : PEREZ	String	Apellido Paterno del Asegurado
MaternoAsegurado	Apellido Materno que se desea consultar. Ejemplo : M	String	Apellido Paterno del Asegurado
NombreAsegurado	Nombre que se desea consultar. Ejemplo : “vació”	String	Nombre(s) del Asegurado

Tabla 4.1: Parámetros de entrada para la operación 1

Nombre	Descripción	Tipo	Valores
Entidad_Codigo	Código de la EPS a quien se realiza la consulta .Ejemplo 060027ª	String	Código de Registro de la EPS proporcionada por la SEPS.
EVinculada_Codigo	Código de Registro de la Entidad Vinculada. Ejemplo 980005C	String	Código de Registro de la Entidad Vinculada
EVinculada_Ruc	RUC de la Entidad Vinculada. Ejemplo : 20100162742	String	RUC de la Entidad Vinculada
Asegurado_Codigo	Código que se desea consultar. Ejemplo : 06000201	String	Código del asegurado
Asegurado_Plan	Código del Plan que se desea consultar. Ejemplo : 16	String	Código del Plan de Salud del asegurado.

Tabla 4.2: Parámetros de entrada para la operación 2

Nombre	Descripción	Tipo	Valores
Entidad_Codigo	Código de la EPS a quien se realiza la consulta .Ejemplo 060027A	String	Código de Registro de la EPS proporcionada por la SEPS.
EVinculada_Codigo	Código de Registro de la Entidad Vinculada. Ejemplo : 980005C	String	Código de Registro de la Entidad Vinculada
EVinculada_Ruc	RUC de la Entidad Vinculada. Ejemplo : 20100162742	String	RUC de la Entidad Vinculada
Asegurado_Codigo	Código que se desea consultar. Ejemplo : 06000201	String	Código del asegurado
Asegurado_Plan	Código del Plan que se desea consultar. Ejemplo : 16	String	Código del Plan de Salud del asegurado.
CodigoCobertura	Código de la Cobertura. Ejemplo : 47	String	Ver Anexo II , Tabla 2.1
Servicio	Servicio brindado en la Cobertura. Ejemplo. : ZU	String	Ver Anexo II , Tabla 2.2
CopagoFijo	Monto del Copago Fijo. Ejemplo.: 000051.00	String	Valor en formato 999999.99 del Copago Fijo
Copagovariable	Porcentaje de Copago Variable. Ejemplo: 085.00	String	Valor en formato 999.99 del Copago Variable

Tabla 4.3: Parámetros de entrada para la operación 3

Nombre	Descripción	Valores
Entidad_Codigo	Código de la EPS a quien se realiza la consulta .Ejemplo 060027ª	Código de Registro de la EPS proporcionada por la SEPS.
EVinculada_Codigo	Código de Registro de la Entidad Vinculada. Ejemplo 980005C	Código de Registro de la Entidad Vinculada
EVinculada_Ruc	RUC de la Entidad Vinculada. Ejemplo : 20100162742	RUC de la Entidad Vinculada
Asegurado_Codigo	Código que se desea consultar. Ejemplo : 06000201	Código del asegurado

Tabla 4.4: Parámetros de entrada para la operación 4

Nombre	Descripción	Valores
Entidad_Codigo	Código de la EPS a quien se realiza la consulta .Ejemplo 060027ª	Código de Registro de la EPS proporcionada por la SEPS.
EVinculada_Codigo	Código de Registro de la Entidad Vinculada. Ejemplo 980005C	Código de Registro de la Entidad Vinculada
EVinculada_Ruc	RUC de la Entidad Vinculada. Ejemplo : 20100162742	RUC de la Entidad Vinculada
Asegurado_Codigo	Código que se desea consultar. Ejemplo : 06000201	Código del asegurado
Tipo	Código del Tipo de Observación .	1: Datos Adicionales 2: Observaciones

Tabla 4.5: Parámetros de entrada para la operación 5

En el Anexo 05 se puede observar un ejemplo de la funcionalidad de los Servicios Web, este ejemplo muestra una trama de respuesta a la consulta que se le realiza a la operación respectiva del Servicio Web.

CAPITULO V

DESCRIPCIÓN DE LA SOLUCIÓN TECNOLÓGICA

En este capitulo se presenta el sistema de información para la resolución del problema.

5.1 Modelo del Negocio del Prototipo

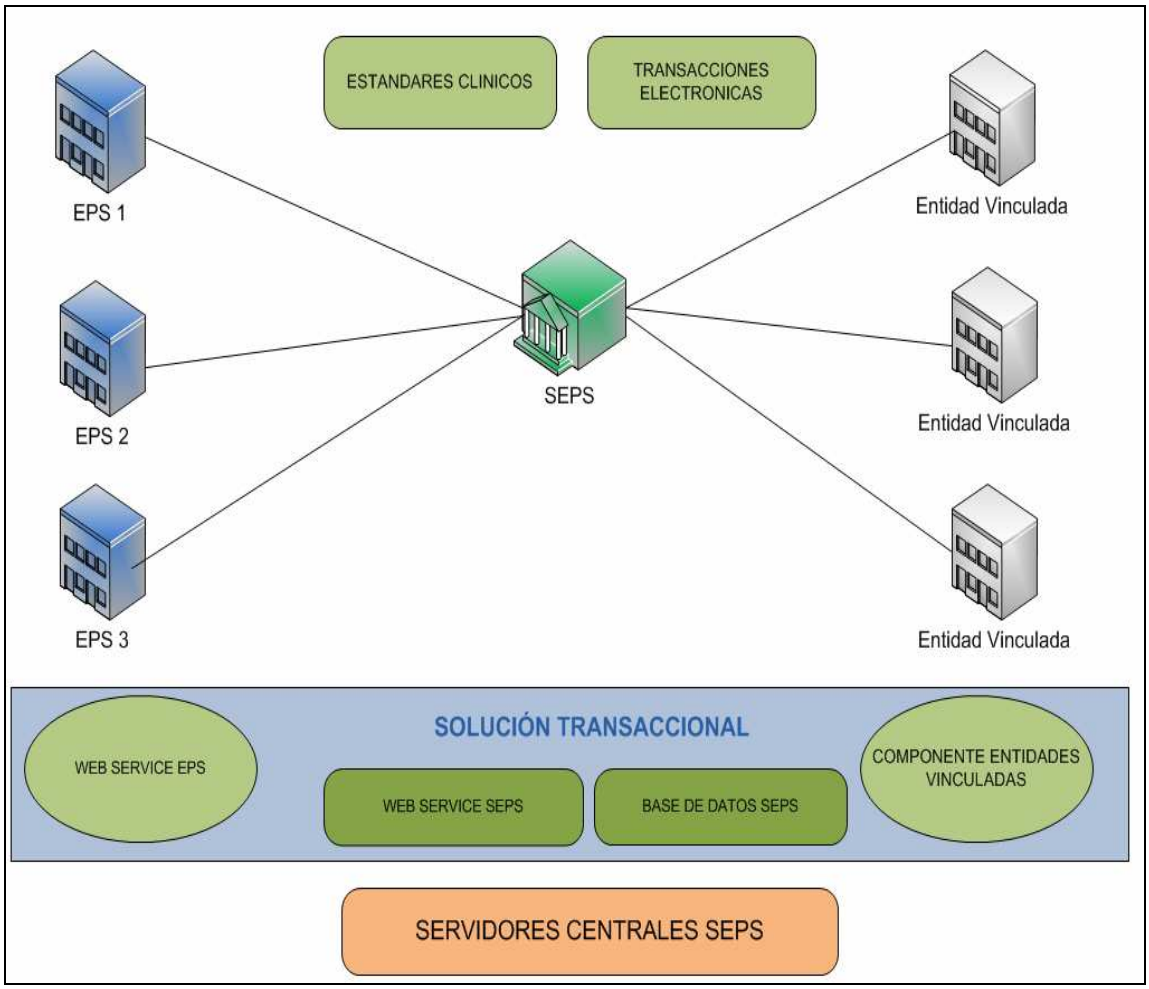

Figura Nº 5.1: Modelo del Negocio del SITEDS

5.2 Modelo de Flujo del Negocio

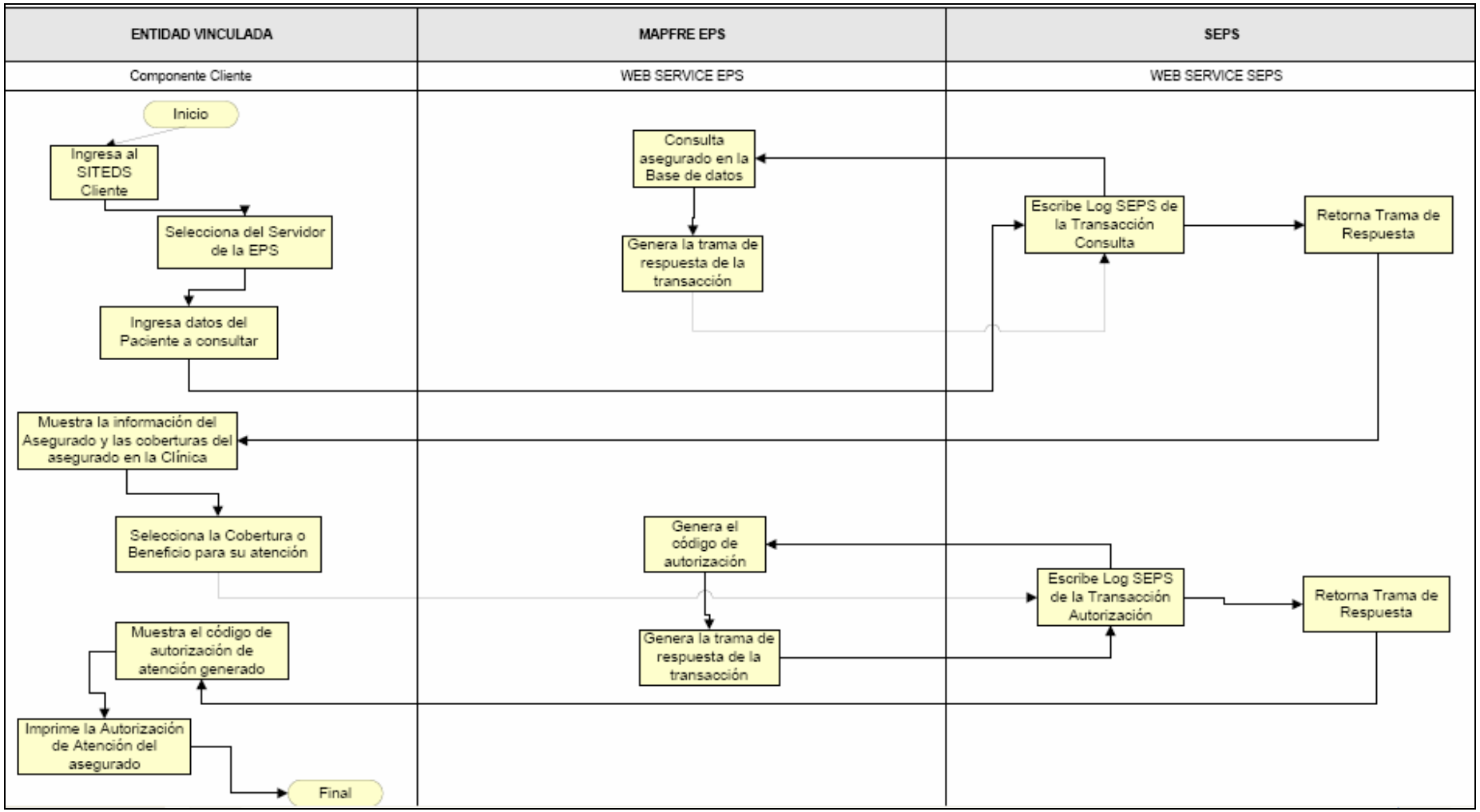

Figura Nº 5.2: Modelo de Flujo del SITEDS

5.3 Diagrama de Actividades

Los diagramas de actividades presentados a continuación, se centran en las operaciones realizadas por el Servicio Web EPS, al ser estos los de mayor relevancia en la solución informática propuesta en la investigación.

- Búsqueda por Nombre del Asegurado
- Búsqueda por Código del Asegurado
- Búsqueda Datos Adicionales del Asegurado
- Búsqueda de Observaciones del asegurado.
- Búsqueda de Foto del Asegurado

5.3.1 Búsqueda por Nombre del Asegurado

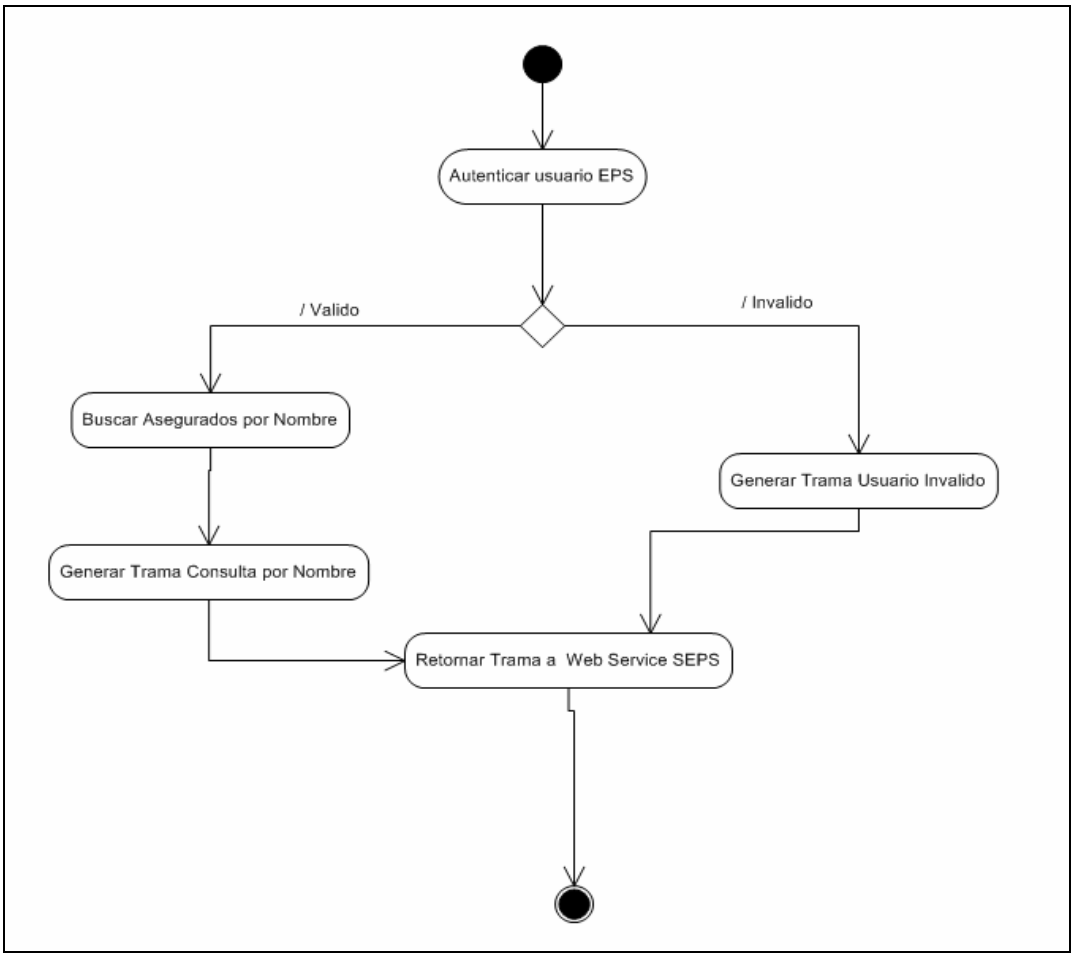

Figura 5.3: Diagrama de Actividad Búsqueda por Nombre en WS EPS

5.3.2 Búsqueda por Código del Asegurado

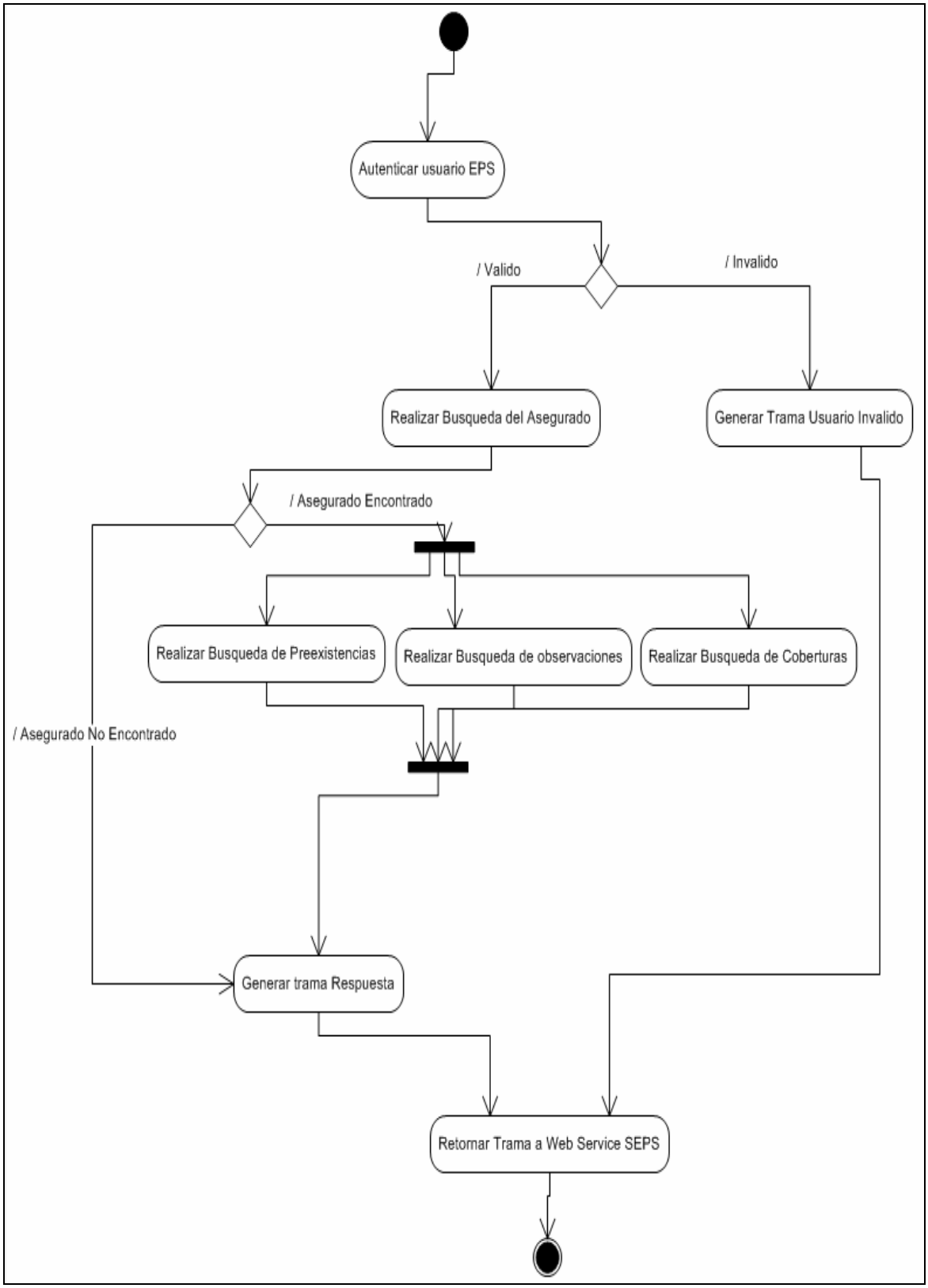

Figura 5.4: Diagrama de Actividad Búsqueda por Código de Asegurado en WS EPS

5.3.3 Búsqueda Datos Adicionales del Asegurado

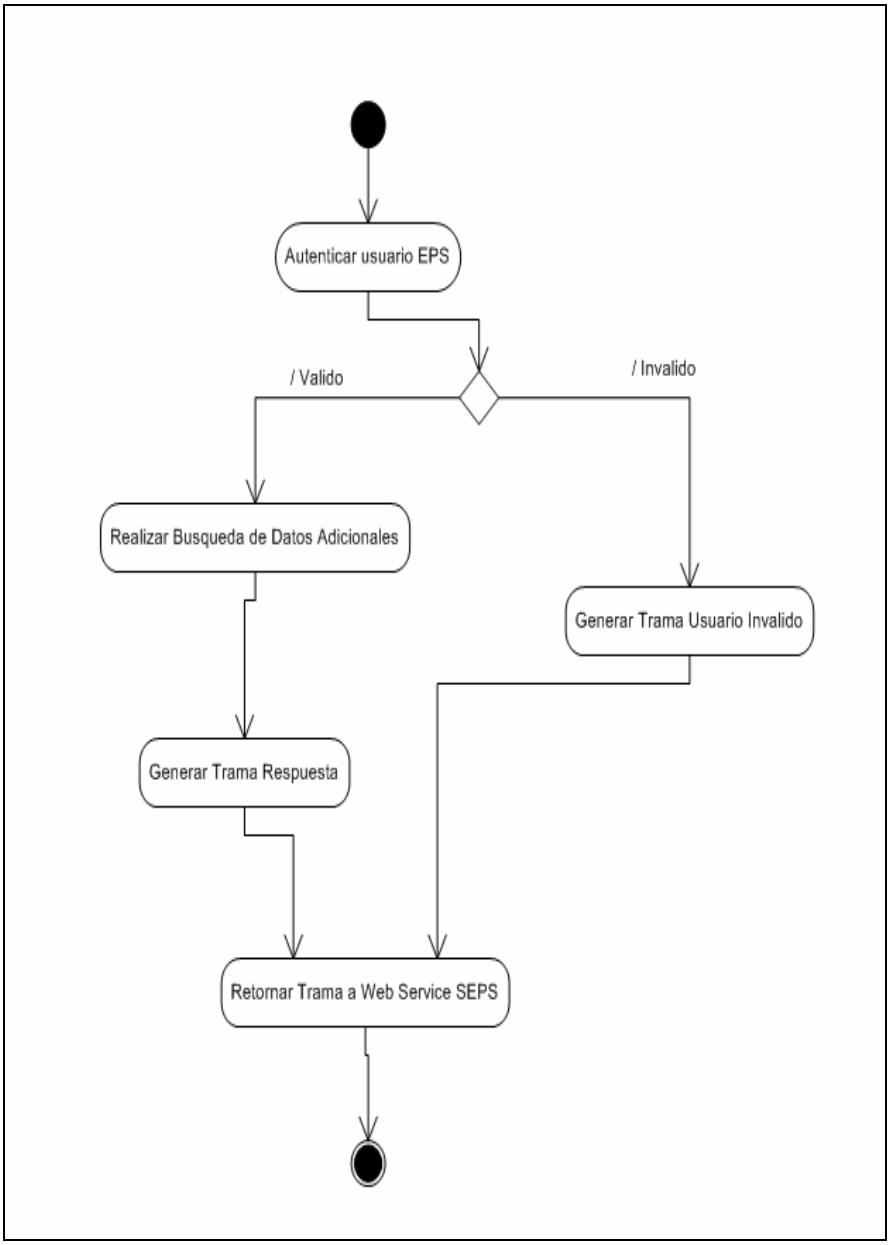

Figura 5.5: Diagrama de Actividad Búsqueda Datos Adicionales del Asegurado en WS EPS

5.3.4 Búsqueda de Observaciones del asegurado.

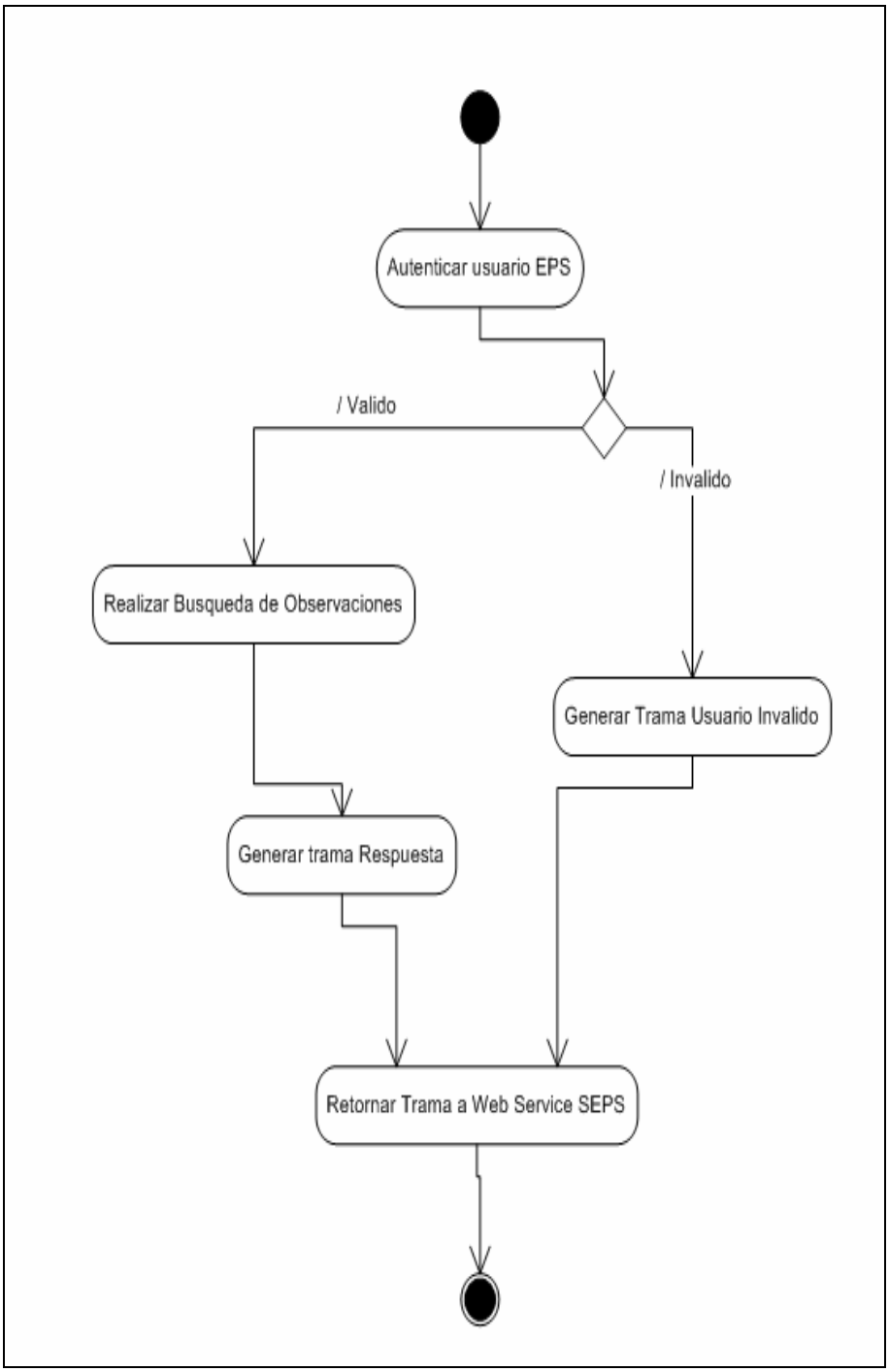

Figura 5.6: Diagrama de Actividad Búsqueda Observaciones del Asegurado en WS EPS

5.3.5 Búsqueda de Foto del Asegurado

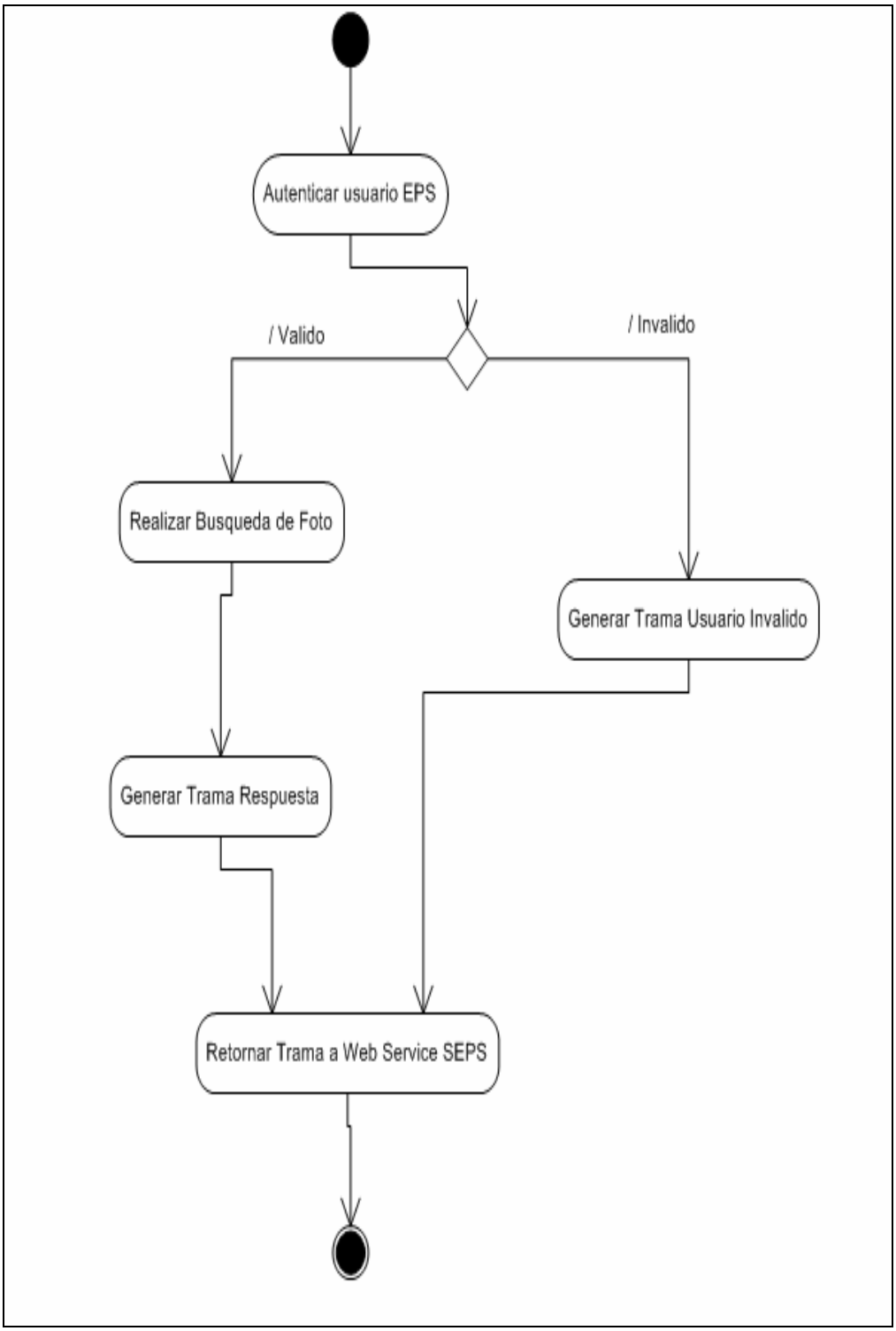

Figura 5.7: Diagrama de Actividad Búsqueda de Foto de Asegurado en WS EPS

5.4 Diagrama de los casos de uso del Sistema

El objetivo de este apartado es la definición de los diagramas de caso de uso del Sistema.

5.4.1 Actor

En este apartado se describen el actor que intervienen en los procesos que tienen lugar en el sistema, con los casos de uso en los que interactúan.

5.4.1.1 Personal de Admisión EV

Actor	Personal de Admisión EV
Descripción	Personal que labora en el área de admisión o atención al paciente de las Entidades Vinculadas.
Casos de Uso	CU-DA CU-GA

5.4.2 Casos de Uso

En este apartado se describen los casos de uso del sistema.

Caso de Uso	Consultar Datos del Asegurado
ID	CU-DA
Descripción	El sistema muestra los datos del asegurado de la EPS que solicita la prestación de salud.

Caso de Uso	Autorizar Atención del Asegurado
ID	CU-GA

Descripción	El actor selecciona la cobertura o beneficio para la atención del asegurado y solicita la autorización.

Caso de Uso	Solicitar Lista de Asegurados
ID	CU-SL
Descripción	El usuario ingresa los criterios de búsqueda de datos del asegurado para obtener el listado de asegurados que coinciden con los datos ingresados.

Caso de Uso	Registrar Log de Transacción
ID	CU-LT
Descripción	El sistema almacena el registro de la orden de atención.

Caso de Uso	Autenticar Usuario EV
ID	CU-AE
Descripción	El sistema valida la información del usuario que desea realizar la transacción.

Caso de Uso	Enrutar Consulta
ID	CU-EC
Descripción	El Servicio Web SEPS recibe el tipo de consulta con sus respectivos parámetros y lo enruta al

	servicio Web EPS y a su respectivo método Web.

Caso de Uso	Almacenar Log de la transacción
ID	CU-SL
Descripción	El sistema almacena todas las transacciones realizadas por el servicio Web SEPS.

Caso de Uso	Atender Consulta
ID	CU-AC
Descripción	El sistema responde a la consulta , realizando la búsqueda en la base de datos de la EPS

Caso de Uso	Generar Trama
ID	CU-GT
Descripción	El sistema genera la trama de respuesta en función a la consulta atendida.

5.4.3 Diagrama del Caso de uso principal

Dentro de este diagrama figura un único caso de uso que representa el sistema y todos los actores que interaccionan con él.

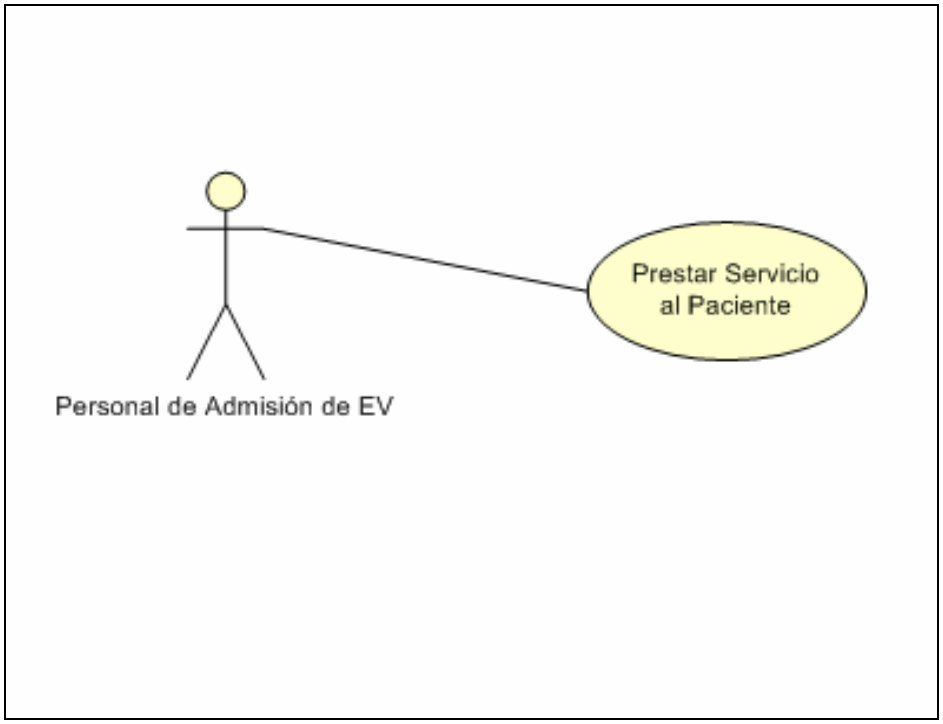

Figura 5.8: Diagrama de caso de uso principal

5.4.4 Diagrama de casos de uso

El diagrama siguiente muestra la relación entre los actores y los casos de uso del sistema. Representa la funcionalidad que ofrece el sistema en lo que se refiere a su interacción externa.

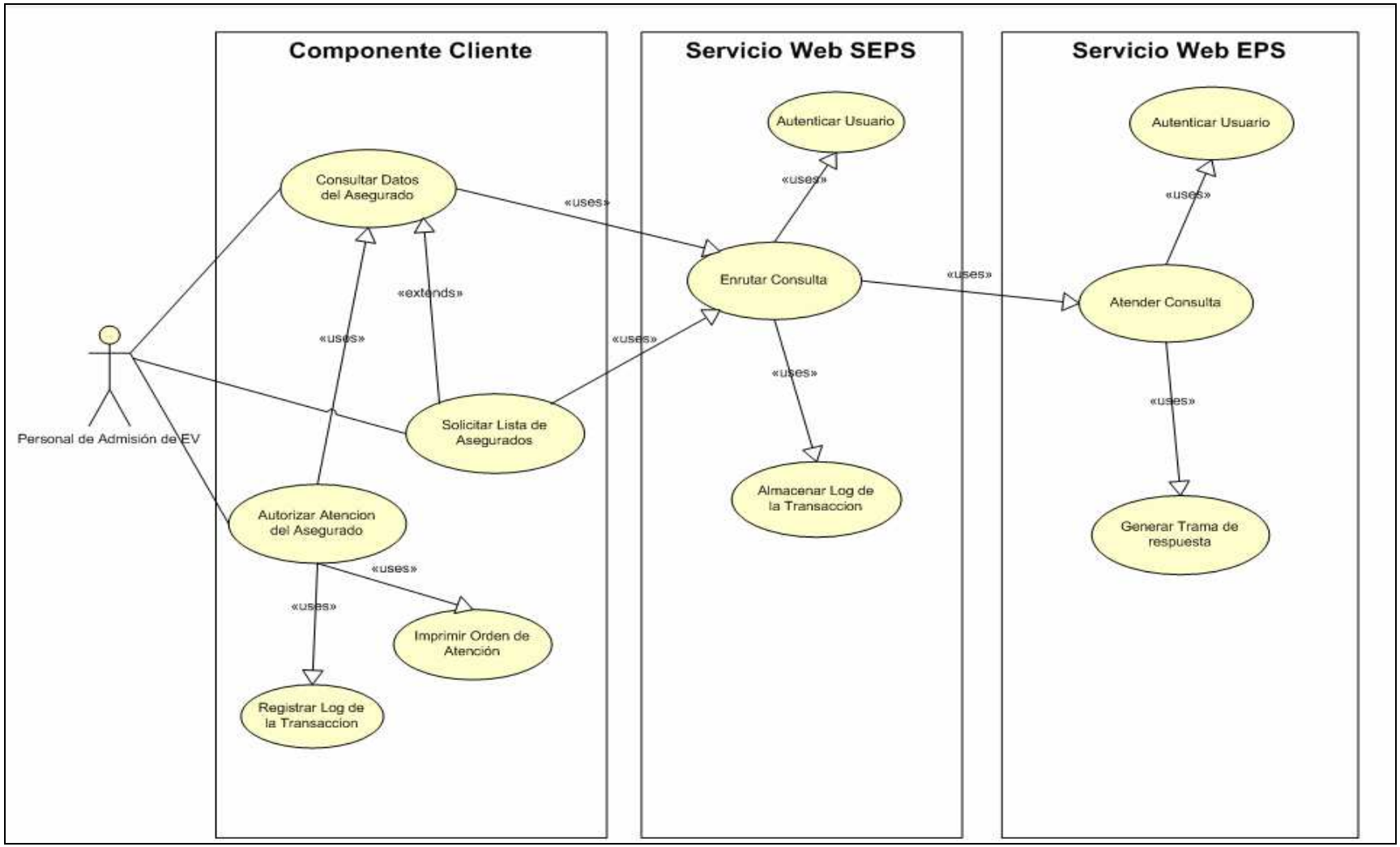

Figura 5.9: Diagrama de casos de uso

A continuación se describen la especificación de los casos de uso más relevantes del sistema:

CU1: Consultar Datos del Asegurado

- **Actores**

  Personal de Admisión de EV

- **Precondiciones**

  Se requiere un inicio de sesión en el sistema.

  Se selecciona el Servidor de MAPFRE EPS.

  Se conoce el código del asegurado.

- **Flujo Básico**

  1.-El actor selecciona en la interface la pestaña de búsqueda por Código.

  2.-El actor ingresa en el campo código de la interface, el código del asegurado solicitante de atención y presiona el botón de “Buscar”

  3.-El sistema procesa la consulta.

  4.-El sistema muestra en los campos correspondientes la siguiente información:

  4.1 Datos del asegurado : Apellidos y Nombres ,parentesco ,sexo ,edad, documento de identidad , inicio de vigencia , fin de vigencia ,estado, fecha nacimiento ,número del plan de salud, numero del carné

  4.2 Datos del titular: Apellidos y Nombres, Numero del Contrato, Empresa Empleadora, RUC de la Empresa, Tipo de Afiliación, Moneda, Fecha de Inclusión

5.- El sistema activa el botón de “Pre existencia” en caso el asegurado tenga alguna preexistencia en su contrato.

6.-El Sistema activa el botón de “Observaciones” en caso el asegurado presente alguna observación ya sea del mismo asegurado y/o del titular. También puede existir alguna observación adicional para la atención.

7.- El sistema muestra el listado de Coberturas del paciente en la entidad vinculada: Código de Cobertura, Descripción de la Cobertura, servicio, copago Fijo, copago variable, Fin de carencia, Parentesco, Sexo y Observación de la cobertura.

- **Flujo Alterno**

  En el Paso 4.

  Si no se ubica al asegurado el sistema genera un mensaje informativo indicando que no existe asegurado con el código indicado.

  En el paso 7.

  Si el asegurado no tiene coberturas en el establecimiento de salud, el sistema genera un mensaje de información indicando que no tiene coberturas.

- **Postcondiciones**

  Ninguna

CU2: Autorizar Atención del Asegurado

- **Actores**

  Personal de Admisión de EV

- **Precondiciones**

Se requiere contar que le asegurado cuente con coberturas en el establecimiento de Salud.

- **Flujo Básico**

  1.-El actor selecciona la cobertura para la atención del asegurado.

  2.-El actor ingresa en el campo código de la interface, el código del asegurado solicitante de atención y presiona el botón de “Buscar”

  3.-El sistema procesa la consulta.

  4.-El sistema muestra en los campos correspondientes la siguiente información:

  4.1 Datos del asegurado : Apellidos y Nombres ,parentesco ,sexo ,edad, documento de identidad , inicio de vigencia , fin de vigencia ,estado, fecha nacimiento ,número del plan de salud, numero del carné

  4.2 Datos del titular: Apellidos y Nombres, Numero del Contrato, Empresa Empleadora, RUC de la Empresa, Tipo de Afiliación, Moneda, Fecha de Inclusión

  5.- El sistema activa el botón de “Pre existencia” en caso el asegurado tenga alguna preexistencia en su contrato.

  6.-El Sistema activa el botón de “Observaciones” en caso el asegurado presente alguna observación ya sea del mismo asegurado y/o del titular. También puede existir alguna observación adicional para la atención.

  7.- El sistema muestra el listado de Coberturas del paciente en la entidad vinculada: Código de Cobertura, Descripción de la Cobertura, servicio, copago Fijo, copago variable, Fin de carencia, Parentesco, Sexo y Observación de la cobertura.

- **Flujo Alterno**

  En el paso 7.

  Si el asegurado no tiene coberturas en el establecimiento de salud, el sistema genera un mensaje de información indicando que no tiene coberturas.

- **Postcondiciones**

  Ninguna

CU3: Solicitar lista de Asegurados

- **Actores**
  Personal de Admisión de EV

- **Precondiciones**

  No se conoce el código del asegurado pero si los apellidos y nombres.

- **Flujo Básico**

  1.-El actor ingresa en el campo apellido paterno de la interface el apellido paterno del asegurado y/o ingresa en el campo correspondiente el apellido materno del asegurado y/o el nombre del asegurado.

  2.-El actor presiona el botón de “Buscar”

  3.-El sistema procesa la consulta.

  4.-El sistema muestra como resultado un listado de asegurados coincidentes con los criterios de búsqueda, con la siguiente información del asegurado: Apellidos Paterno, Apellido Materno, Nombres, código del asegurado, número de plan de salud, sexo, fecha de nacimiento, estado y tipo de producto.

  5.- El actor hace doble clic sobre el asegurado a seleccionar.

- **Flujo Alterno**

  En el paso 4.

  Si no existen datos coincidentes con el criterio de búsqueda, el sistema genera un mensaje de información indicando que no se encontraron datos coincidentes.

- **Postcondiciones**

  Ninguna

- **Diseño de Capas del Servicio Web EPS**

La Arquitectura del subsistema de Servicio Web EPS va a estar formado por componentes agrupados en tres niveles o capas:

- **Capa de presentación:** Contiene las interfases de usuario. Muestran y capturan la información que el usuario ingresa en el subsistema. Esta capa se comunica con la capa de lógica del negocio y con el subsistema de seguridad.
- **Capa de lógica del negocio:** Contiene la lógica del subsistema. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de acceso a datos, para solicitar al gestor de la base de datos para almacenar o recuperar datos de él.
- **Capa de acceso a datos:** Permite obtener información de la base de datos del subsistema; también se podrá registrar y eliminar información de la base de datos. Esta capa se comunica con la capa de lógica del negocio y con la base de datos del subsistema.

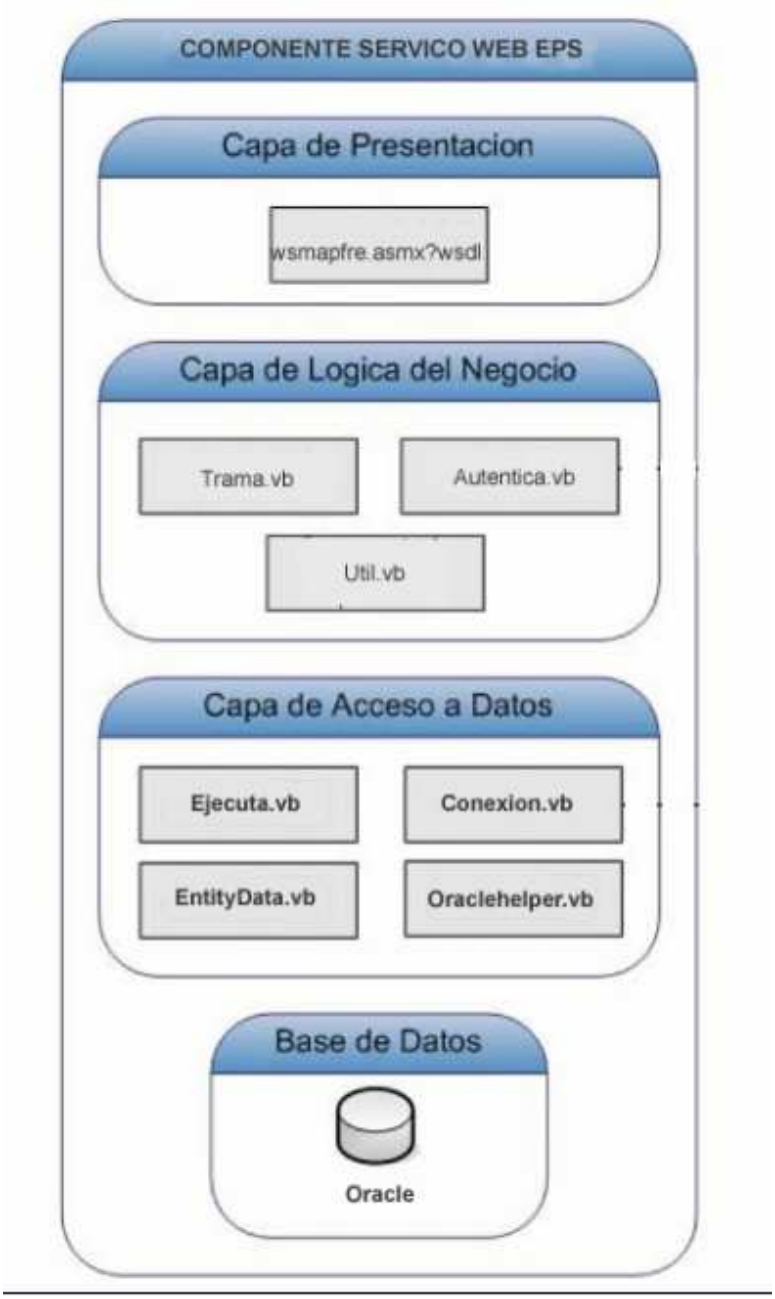

Figura N°5.10: Arquitectura del Servicio Web EPS

A continuación se detalla el diseño de la Capas del Web Service EPS.

- Capa de Presentación , esta formada por :
  - **wsmapfre.asmx**: Servicio Web que contiene todos los servicios requeridos por el Servicio Web SEPS.
- Capa de Lógica de negocio , esta formada por las clases :
  - **Trama**: Se encarga del manejo de la generación de la trama.

- **Autentifica**: Se encarga de realizar la autenticación del usuario.
- **Útil**: se encarga de dar formato a la trama.

- Capa de Acceso de Datos , esta formada por las clases :
  - **Ejecuta**: el encargado de brindar las funciones para obtener registros de la Base de datos.
  - **Conexion**: Se encarga de obtener del archivo de configuración la cadena de conexión.
  - **EntityData**: Se encarga de interactuar con la clase OracleHelper.
  - **OracleHelper**: tiene las siguientes funciones ExecuteDataTable, ExecuteNonQuery, ExecuteDataset, etc.

- **Diagrama de Clases**

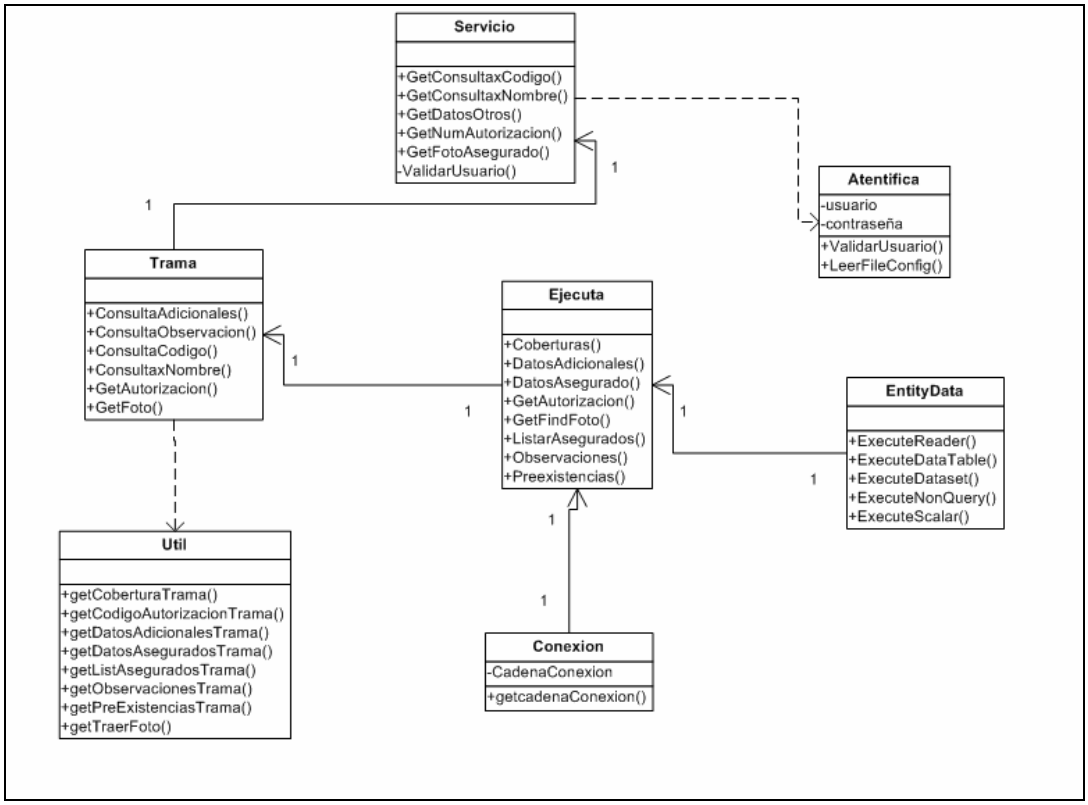

Figura 5.11: Diagrama de Clases Servicio Web EPS

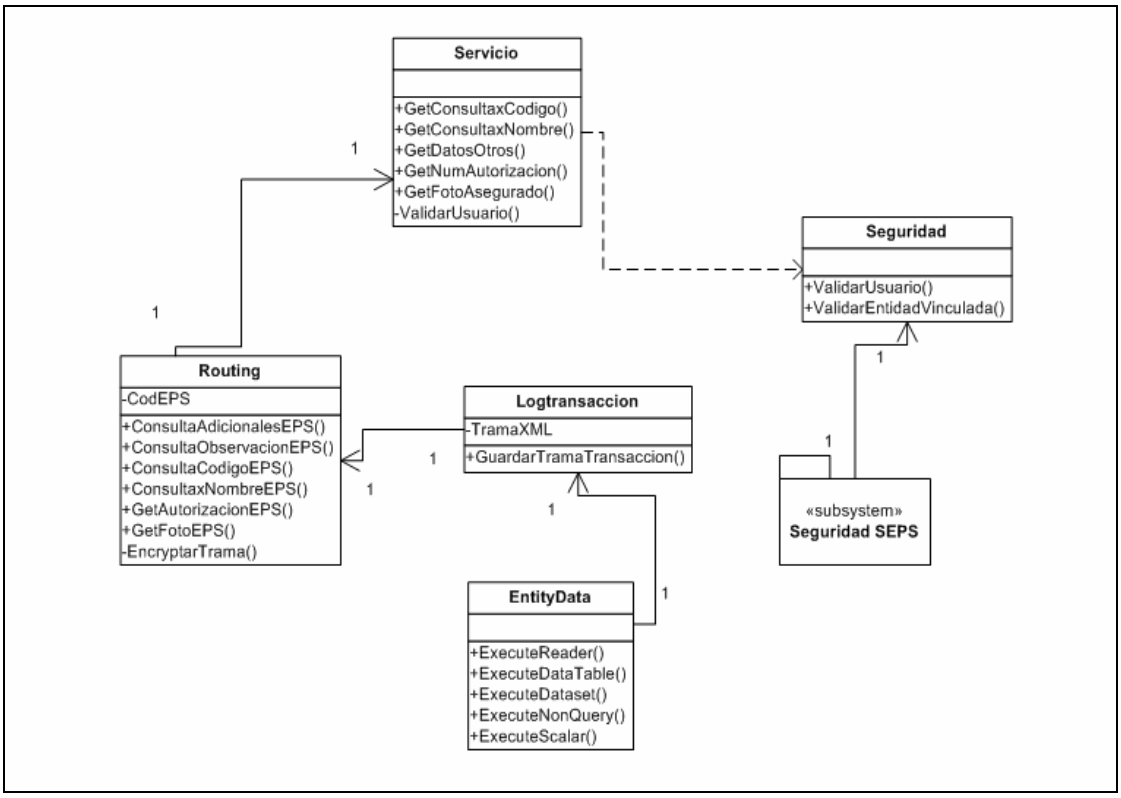

Figura 5.12. Diagrama de Clases Servicio Web SEPS

- **Diagrama de Secuencia**

CU-DA: Consultar datos del Asegurado

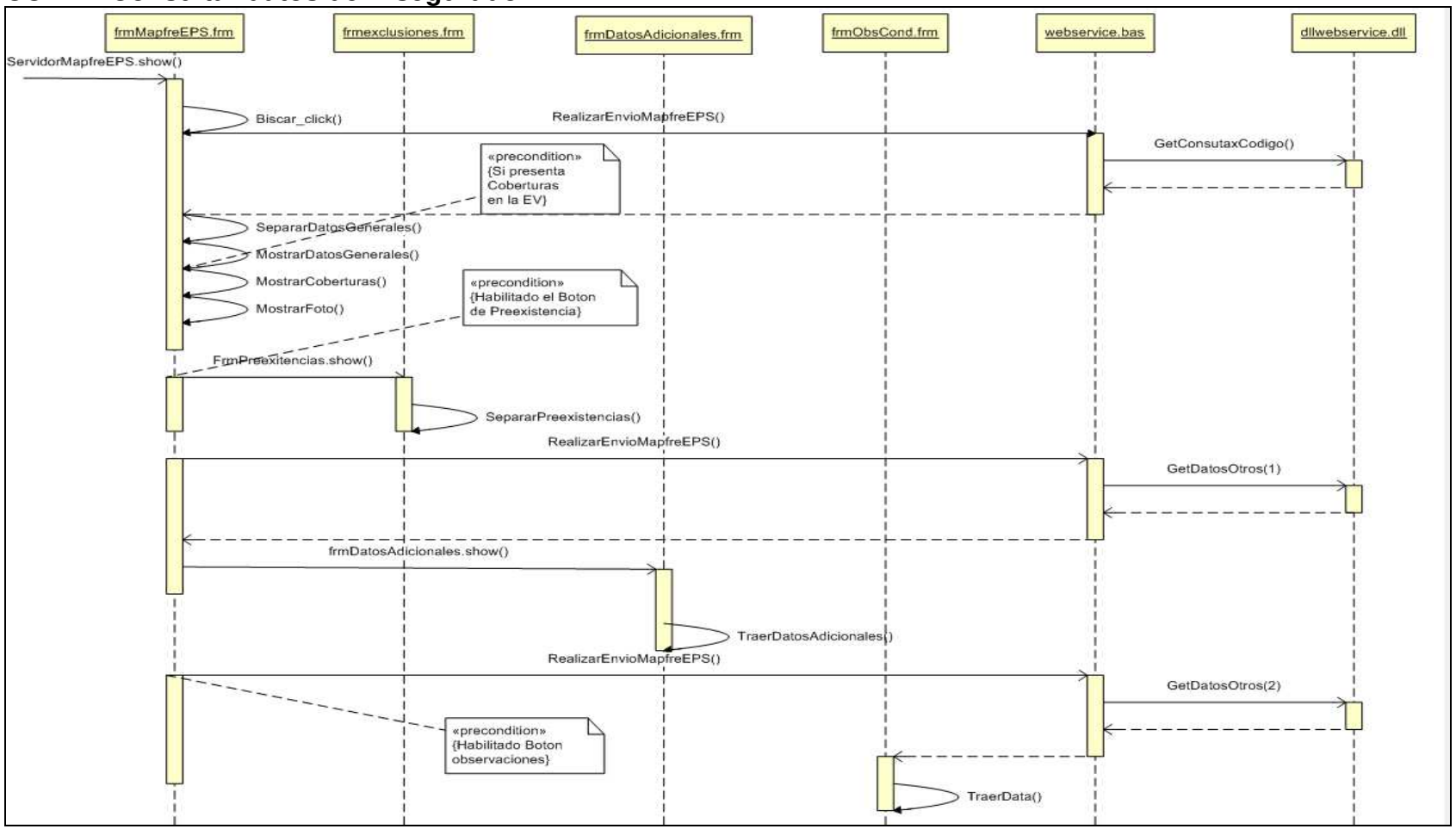

Figura Nº 5.13: Diagrama de Secuencias de Caso de Uso CU-DA

CU-EC: Enrutar Consulta

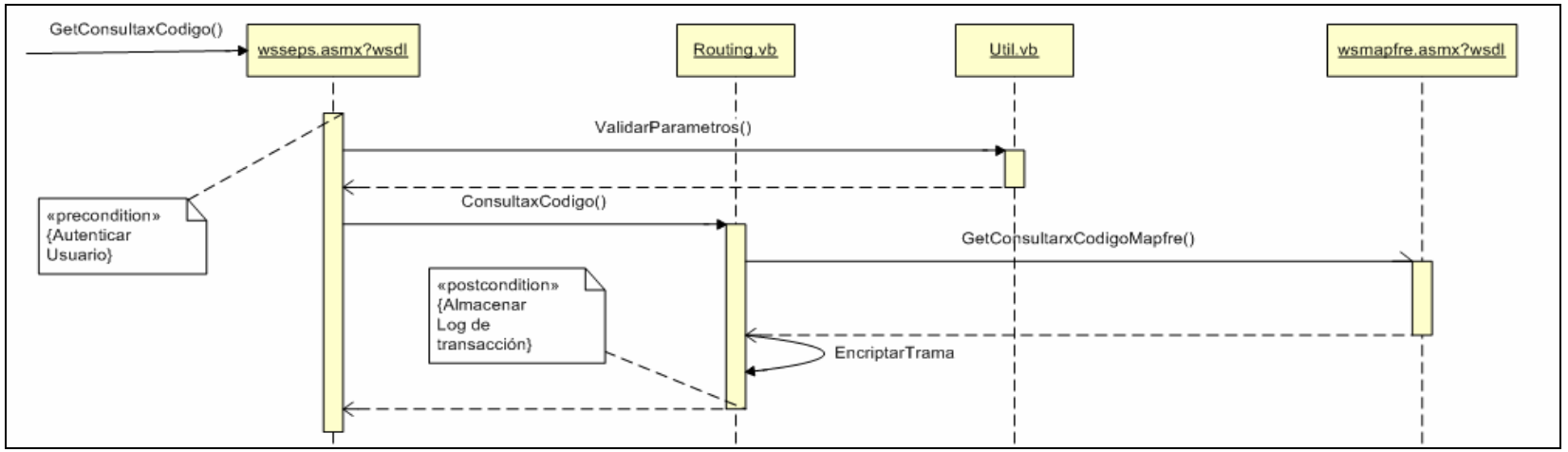

Figura Nº 5.14: Diagrama de Secuencias de Caso de Uso CU-EC

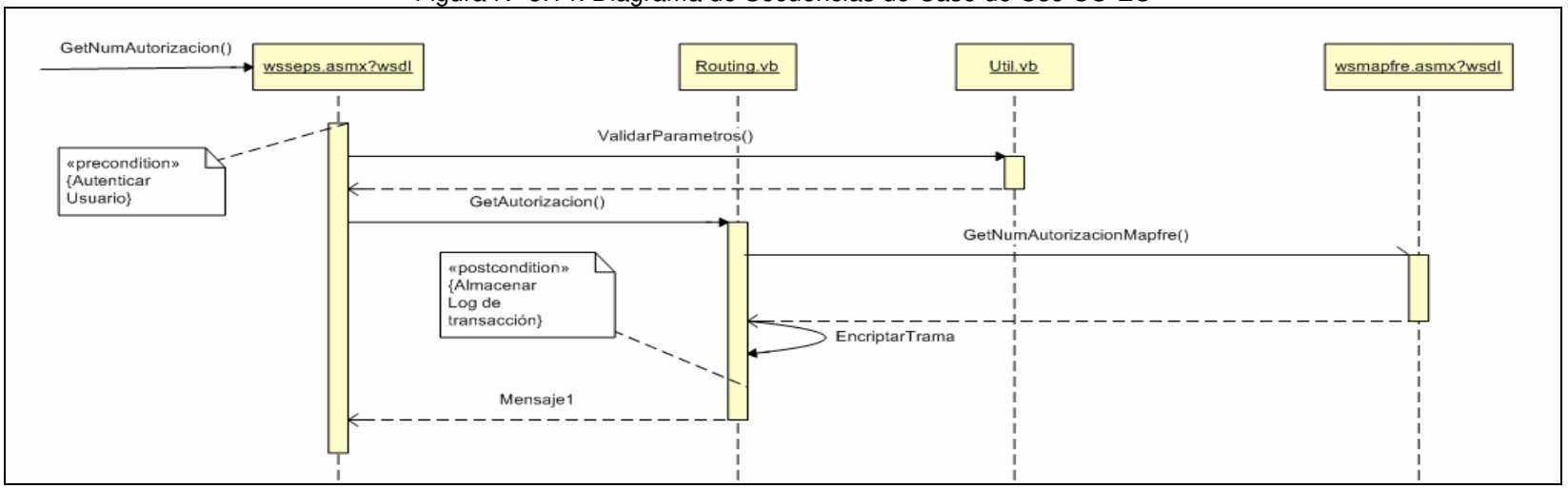

Figura Nº 5.15: Diagrama de Secuencias de Caso de Uso CU-EC

CU-AC: Atender Consulta (Consulta por Código)

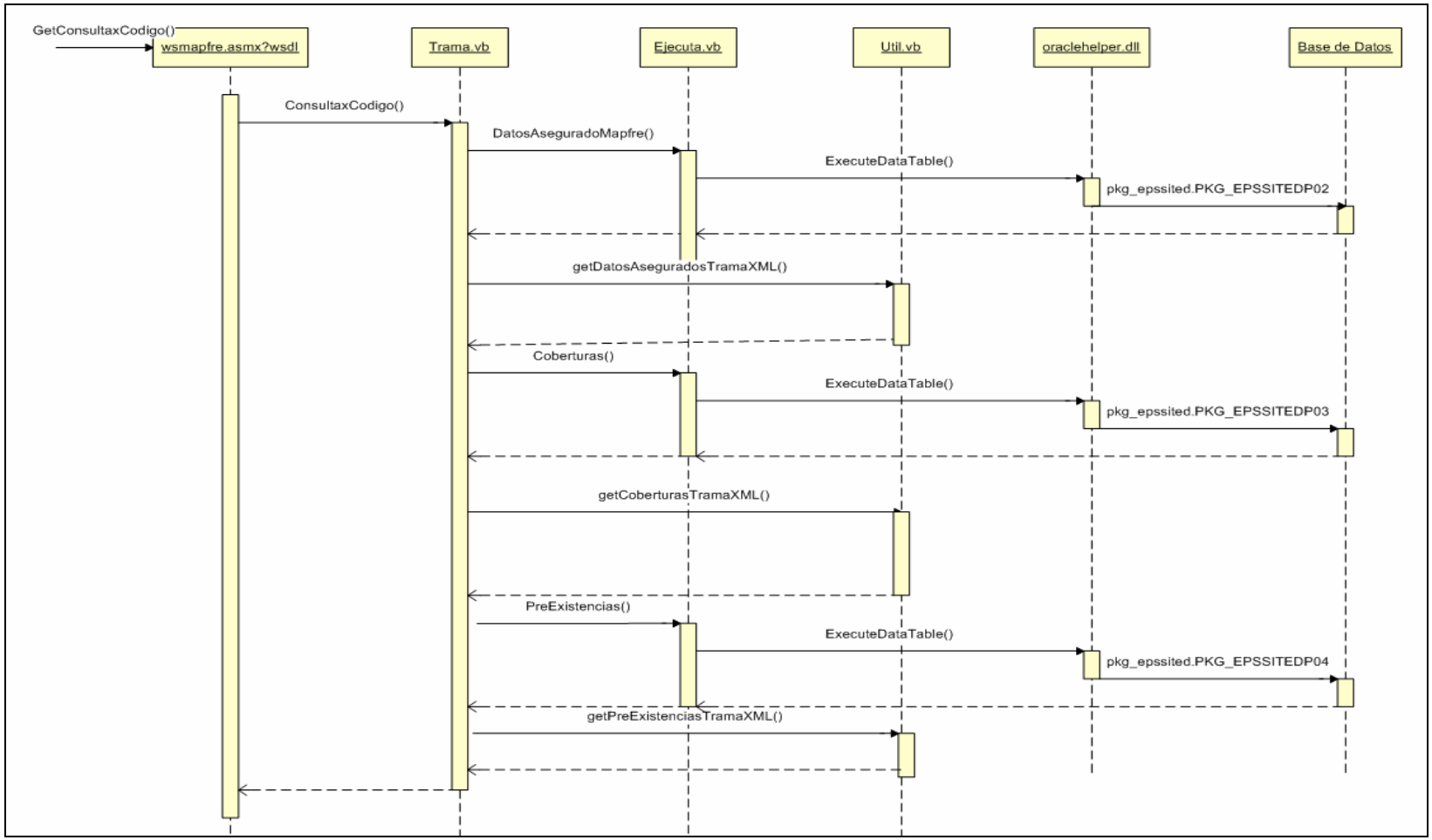

Figura Nº 5.16: Diagrama de Secuencias de Caso de Uso CU-AC

CU-AC: Atender Consulta (Número de Autorización)

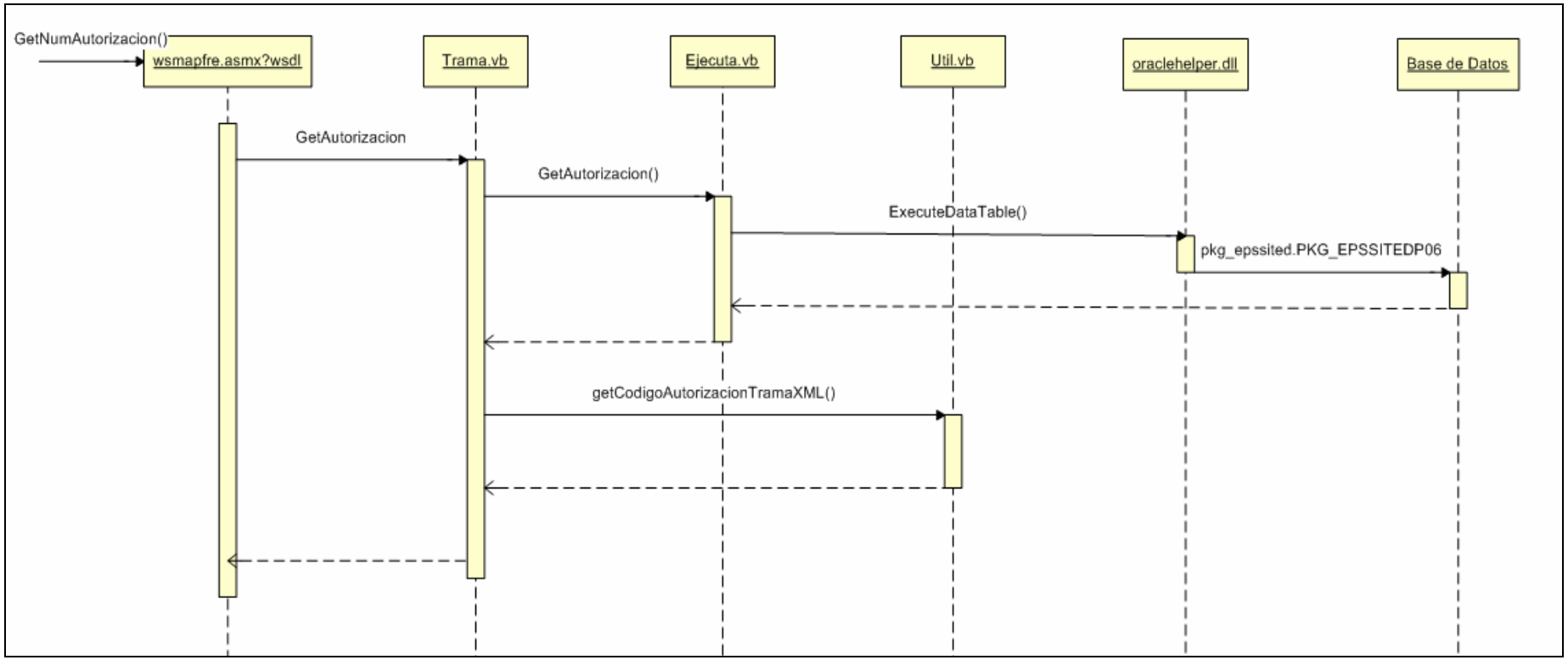

Figura Nº 5.17: Diagrama de Secuencias de Caso de Uso CU-AC

- **Diagrama de Distribución**

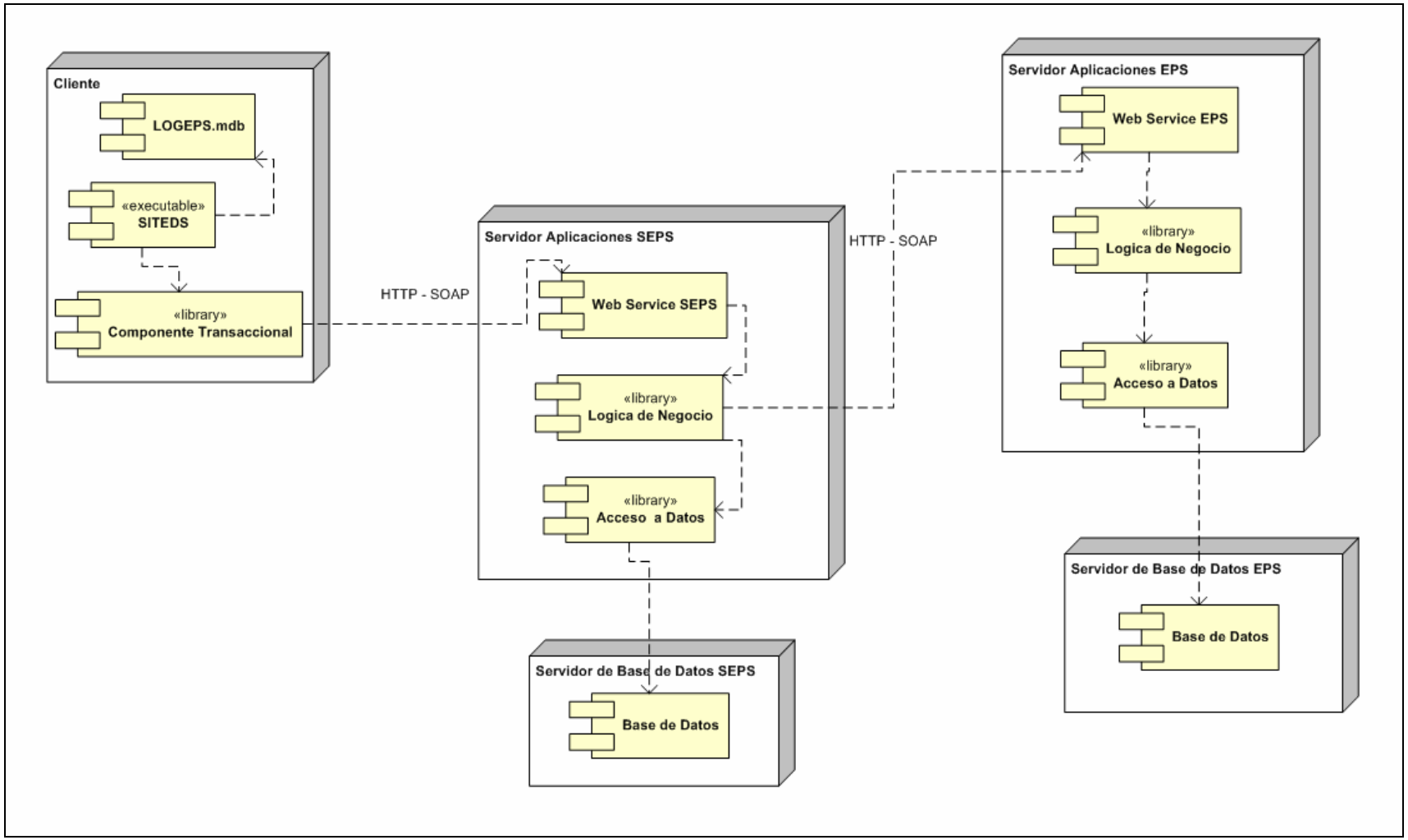

Figura 5.18: Diagrama de Distribución de la Solución Informática

- **Consideraciones sobre el ambiente de desarrollo**

La Oficina de Sistemas de Información utiliza fuertemente la plataforma .Net, junto con todas las herramientas relacionadas con ella, siendo el lenguaje de desarrollo oficial en la institución el Visual Basic , por lo que , como herramienta de desarrollo se empleo el Visual Studio 2005.

Además, se le dio prioridad a la utilización de herramientas Microsoft, tales como IIS, WSE 3.0, etc.

Los Procedimientos Almacenados que responden a las consultas, localizados en la Base de Datos de la EPS, se realizaron en coordinación con la EPS, siguiendo la estructura estándar del SITEDS.

El Sistema de Seguridad interno de la SEPS es el encargado de gestionar los usuarios en el servicio web SEPS.

5 Modulo o Subsistemas de la Solución

El Sistema esta compuesto de tres Módulos:

- **Componente Servicio Web SEPS**, instalado en el Nodo de la SEPS, que administra y enruta los mensajes o transacciones electrónicas.
- **Componente Servicio Web EPS**, instalado en la EPS, que recibe las transacciones y las responde luego de interactuar con su Base de Datos.
- **Componente Cliente Clínicas**, instalado en el proveedor de servicio (Entidades Vinculadas), que genera las transacciones a partir de los sistemas informáticos propios de las Clínicas.

- **Requerimiento mínimo de Hardware y Software**

Cliente:

- CPU: Intel P3 a 800 MHz.
- Memoria: 512 MB
- SO: Windows XP Profesional
- .NET Framework 2.0

Servidor de aplicaciones:

- CPU: Intel P4 a 2,8 GHz.
- Memoria: 1024 MB
- SO: Windows Server 2003 Standard Edition
- .NET Framework 2.0

Servidor de base de datos:

- CPU: Intel P4 a 2,8 GHz,
- Memoria: 1024 MB
- SO: Windows Server 2003 Standard Edition
- Base de datos: Oracle 9i

Internet:

- 2 Mbps de Ancho de Banda para la EPS y SEPS.

CAPITULO VI

6.1 Conclusiones y futuros Trabajos

- El diseño de sistemas distribuidos implica diseñar su arquitectura considerando características como compartir recursos de hardware y/o software, concurrencia, escalabilidad, tolerancia a fallas y transparencia. Típicamente, dichas características tienen que ser contempladas desde un inicio, lo cual implica realizar varias consideraciones para cada una de ellas. Como consecuencia, los sistemas distribuidos tienden a ser complejos, la seguridad es difícil de manejar, el mantenimiento requiere un mayor esfuerzo y el tiempo de respuesta puede variar de acuerdo a las condiciones de la red.

- La presente investigación se enfoca a la idea de partir de una aplicación distribuida existente para obtener una solución basada en Servicios Web de la misma sin necesidad de realizar cambios en la lógica de negocio. Como consecuencia, los desarrolladores no necesitan invertir tiempo para agregar o realizar cambios a los módulos, debido a que se reduce el proceso de desarrollo.

- Con la implantación de este modelo de sistema distribuida, basado en Servicio Web, se logra que las transacciones entre los componentes SITEDS, sean adaptables, extendibles y personalizables.

- La solución propuesta permite a distintas aplicaciones, de diferentes orígenes, de las Entidades Vinculadas consumir la información del SITEDS sin necesidad de escribir programas costosos o emplear librerías intermedias, y esto es posible, porque la comunicación se hace con XML y por el protocolo estándar HTTP.

- Al conseguir establecer comunicación con sistemas heterogéneos, las Entidades Vinculadas podrán registrar en sus Base de Datos sus transacciones (Ordenes de atención generadas) realizadas en el SITEDS, en sus bases de datos internas y permitirle a la entidad de salud tener una administración de las atenciones realizadas a los asegurados de las EPS.

- Con la Implantación de esta solución se conseguirá eliminar la dependencia del mantenimiento del sistema que actualmente existe sobre un determinado profesional de informática, puesto que el desarrollo de la solución informática será realizado sobre una plataforma Windows y con un entorno de desarrollo popular en el mercado; por lo que, la disminución en tiempo y costos para los futuros mantenimientos será evidente.

- Con el éxito de la implantación del sistema en el escenario SEPS - MAPFRE EPS, se impulsará que las EPS con mayor número de asegurados cambien a esta solución informática planteada en la presente investigación.

6.1.1 Trabajos Futuros

Emplear la nueva tecnología de desarrollo de Servicio Web propuesta por Microsoft, WCF (*Windows Communication Foundation*); migrando los Servicio Web ASP.NET a WCF sin que afecten a los clientes. Esta nueva tecnología admite más protocolos de servicios Web, estos protocolos adicionales proporcionan más soluciones sofisticadas, como sesiones y transacciones confiables, entre otras.

Implementar en el Servicio Web, métodos que retornen datos binarios de gran volumen, para transmitir documentación necesaria para la atención de un tipo de cobertura especifica, como por ejemplo Imprimir documento de carta de garantía para una intervención quirúrgica.

Elaborar en coordinación con las EPS Rímac y Pacifico un Plan de Migración de los Componentes Servidor EPS a Servicio Web.

La SEPS en coordinación con las EPS debe diseñar Plan de Contingencias para la continuidad del negocio. Implementando sistemas de alta disponibilidad y servidores de contingencia.

Implantar esta arquitectura basada en Servicio Web en los sistemas de Seguros de Salud, como son Rímac Asistencia Medica y Pacífico Seguros, ya que estos actualmente emplean la misma arquitectura de los componentes servidor de las EPS.

BIBLIOGRAFIA

[A&W 2004] Attiya, Hagit and Welch, Jennifer. *Distributed Computing: Fundamentals, Simulations and Advanced Topics*. 2a ed. John Wiley & Sons, Inc. New Jersey. 2004.

[CDK 2001] Coulouris, George; Dollimore, Jean; Kindberg, Tim. *Sistemas Distribuidos: Conceptos y Diseño*. 3a ed. Madrid: Pearson Educación S.A. 2001.

[COMPI 2008] http://undergraduate.csse.uwa.edu.au/units/CITS3213/lectures2-2006/lect6.pdf
Última visita: 20 de mayo del 2008

[COMPII 2008] Juric, M.B., Rozman, I., and Hericko, M., Performance Comparison of CORBA and RMI, Information and Software Technology 42, pp 915-933, 2000. *http://my.execpc.com/~gopalan/misc/compare.html*
Última visita: 20 de mayo del 2008

[COMPIII 2008] http://arcos.inf.uc3m.es/~dsd/ComputacionDistribuida.pdf
Última visita: 20 de mayo del 2008

[COMPIV 2008] http://www.xs4all.nl/~irmen/comp/CORBA_vs_SOAP.html
Última visita: 20 de mayo del 2008

[COMPV 2008] http://cs.nyu.edu/courses/fall03/V22.0480-004/lectures/lect9.pdf
Última visita: 20 de mayo del 2008

[COMPVI 2008] http://www.atsistemas.com/soa.aspx
Última visita: 20 de mayo del 2008

[COMPVII 2008] http://www.capacinet.gob.mx/wb2/eMex/eMex_Web_Services
Última visita: 20 de mayo del 2008

[COMPVIII 2008] http://www.unsj-cuim.edu.ar/portalzonda/congreso/papers/1999/Java_RMI.html#Desventajas
Última visita: 20 de mayo del 2008

[COMPIX 2008] http://www.wikilearning.com/articulo/corba_generalidades-corba_generalidades/12608-1
Última visita: 20 de mayo del 2008

[COMS 2008] Microsoft Corporation. *The Component Object Model Specification.* http://www.microsoft.com/com/resources/specs.asp
Última visita: 17 de mayo del 2008

[G&G 1996] Grunder Holger and Geihs Kurt. "*Reuse and inheritance in distributed object systems"*. Trends in Distributed Systems CORBA and Beyond. Springer-Verlag. Lecture Notes in Computer Science, Vol. 1161, pp. 191-200. 1996. http://citeseer.ist.psu.edu/grunder96reuse.html
Última visita: 16 de mayo del 2008

[GOP 2003] Raj, Gopalan Suresh. "*A Detailed Comparison of CORBA, DCOM and Java/RMI"*. 2003. http://my.execpc.com/~gopalan/misc/compare.html
Última visita: 15 de mayo del 2008

[JAB 1992] Jacobson, Ivar. *Object-Oriented Software Engineering : A Use Case Driven Approach*. Addison-Wesley. 1992.

[JD 2006] Arquitectura orientada a servicios, Guia practica para cuantificar el retorno de inversión, IBM Global Business Service, 2006 ,p 12.

[JRMI 1999] Sun Microsystems, Inc. *Java Remote Method Invocation Architecture and Functional Specification.* Versión 1.3.0. Diciembre, 1999.
http://java.sun.com/j2se/1.3/pdf/rmi-spec-1.3.pdf
Última visita: 20 de mayo del 2008

[KSSRM 2002] Kawsar, F.; Shaikot, S.; Saikat, S.; Razzaque, A.; Mottalib, M. *"An Efficient Dynamic Scheduling Algorithm in Distributed System"*. Proceedings of the 5th International Conference on Computer and Information Technology (ICCIT 2002).
pp. 97-100. Dahka, Bangladesh. Diciembre, 2002.

[MEJ 2007] Christian Iván Mejía Escobar. Herramienta autoconfigurable para el desarrollo de aplicaciones distribuidas de tiempo real flexibles y dinámicas.
http://www.cs.cinvestav.mx/Estudiantes/TesisGraduados/2007/tesisChristiamIvanMejia.pdf
Última visita: 2 de Mayo del 2008

[MOL 2004] Lía Molinari. Arquitecturas Orientadas a Web Services.
http://www.sedici.unlp.edu.ar/search/request.php?id_document=ARG-UNLP-TPG-0000000068&request=request
Última visita: 15 de Abril del 2008

[MSDAS 2006] Microsoft Corporation – Patrones y Buenas Prácticas. "Diseño de Aplicaiones y Servicios".
http://www.microsoft.com/spanish/msdn/arquitectura/das/guias/AppArchCh1.mspx
Última visita: 21 de mayo del 2008

[MSOA 2007] Microsoft Corporation. “La Arquitectura SOA”
download.microsoft.com/download/c/2/c/c2ce8a3a-b4df-4a12-ba18-7e050aef3364/070717-Real_World_SOA.pdf
Última visita: 22 de mayo del 2008

[NET 2006] .NET Framework 2.0 Application Development Foundation , Microsoft Press , 2006 , p 658 –666

[OAS 2006] OASIS Web Services Security (WSS) TC.
http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss
Última visita: 23 de mayo del 2008

[OMG 2004] Object Management Group (OMG). *Common Object Request Broker Architecture (CORBA): Core Specification*. Versión 3.0.3. Marzo, 2004.
http://www.omg.org/cgi-bin/doc?formal/04-03-12
Última visita: 19 de mayo del 2008

[PRP 2006] Puder, Arno; Römer Kay and Pilhofer, Frank. *Distributed Systems Architecture: A Middleware Approach*. Elsevier, 2006.

[SCR002] Understanding SOAP. Kenn Scribner, Mark Stiver. Editorial: SAMS. Marzo 2002.

[SK 2003] Web Services in .NET vs. J2EE, S Kachru - 2003 - lib.ncsu.edu,
http://www.lib.ncsu.edu/theses/available/etd-08172003-193313/unrestricted/etd.pdf
Última visita: 14 de mayo del 2008

[T&V 2002] Tanenbaum, Andrew and Van Steen Maarten. *Distributed Systems: Principles and Paradigms*. Prentice-Hall, Inc. 2002.

[TAN 2003] Tanenbaum, Andrew. *Sistemas Operativos Modernos*. Pearson Educación. 2a edición. 2003.

[V&R 2001] Veríssimo, Paulo and Rodrigues, Luís. *Distributed Systems for System Architects*. Kluwer Academic Publishers. 2001.

[W3C 2004] Consorcio World Wide Web (W3C). “Web Services Architecture”
http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/
Última visita: 23 de mayo del 2008

[WIL 2008] William Brogden, “REST versus SOAP – the REST story”,
http://searchwebservices.techtarget.com/tip/0,289483,sid26_gci1227190,00.html
Última visita: 24 de mayo del 2008

[X12 2008] http://www.x12.org/
Última visita: 24 de mayo del 2008

GLOSARIO

.NET Framework Remoting. .NET Remoting es una interface de programación de aplicación (API) de Microsoft para comunicación de interprocesos lanzado el 2002 con la versión 1.0 del .NET Framework.

ACK. ACKNOWLEDGEMENT (acuse de recibo), en comunicaciones entre computadores, es un mensaje que se envía para confirmar que un mensaje o un conjunto de mensajes han llegado. Si el terminal de destino tiene capacidad para detectar errores, el significado de ACK es "ha llegado y además ha llegado correctamente".

ASCII. El código ASCII (acrónimo inglés de American Standard Code for Information Interchange — Código Estadounidense Estándar para el Intercambio de Información), pronunciado generalmente [áski], es un código de caracteres basado en el alfabeto latino tal como se usa en inglés moderno y en otras lenguas occidentales.

BPM. Se llama Business Process Management a la metodología empresarial cuyo objetivo es mejorar la eficiencia a través de la gestión sistemática de los procesos de negocio (BPR), que se deben modelar, automatizar, integrar, monitorear y optimizar de forma continua.

Cliente-servidor. Esta arquitectura consiste básicamente en que un programa -el cliente- realiza peticiones a otro programa -el servidor- que le da respuesta. Aunque esta idea se puede aplicar a programas que se ejecutan sobre una

sola computadora es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras.

Cluster. El término cluster se aplica a los conjuntos o conglomerados de computadoras construidos mediante la utilización de componentes de hardware comunes y que se comportan como si fuesen una única computadora. Hoy en día juegan un papel importante en la solución de problemas de las ciencias, las ingenierías y del comercio moderno.

Commit. En el contexto de la Ciencia de la computación y la gestión de datos, commit (acción de cometer) se refiere a la idea de hacer que un conjunto de cambios "tentativos, o no permanentes" se conviertan en permanentes. Un uso popular es al final de una transacción de base de datos.

DNS. El Domain Name System (DNS) es una base de datos distribuida y jerárquica que almacena información asociada a nombres de dominio en redes como Internet. Aunque como base de datos el DNS es capaz de asociar diferentes tipos de información a cada nombre, los usos más comunes son la asignación de nombres de dominio a direcciones IP y la localización de los servidores de correo electrónico de cada dominio.

EBCDIC. EBCDIC (Extended Binary Coded Decimal Interchange Code) es un código estándar de 8 bits usado por computadoras mainframe IBM. IBM adaptó el EBCDIC del código de tarjetas perforadas en los años 1960 y lo promulgó como una táctica customer-control cambiando el código estándar ASCII.

Encapsulamiento. En programación modular, y más específicamente en programación orientada a objetos, se denomina encapsulamiento al ocultamiento del estado, es decir, de los datos miembro, de un objeto de

manera que sólo se puede cambiar mediante las operaciones definidas para ese objeto.

EPS. Las Entidades Prestadoras de Salud (EPS) son empresas e instituciones públicas o privadas, distintas a EsSalud, cuyo único fin es prestar servicios de atención de salud, con infraestructura propia y/o de terceros, sujetándose a los controles de la Superintendencia de Entidades Prestadoras de Salud (SEPS).

Escalabilidad. En telecomunicaciones y en ingeniería informática, la escalabilidad es la propiedad deseable de un sistema, una red o un proceso, que indica su habilidad para, o bien manejar el crecimiento continuo de trabajo de manera fluida, o bien para estar preparado para hacerse más grande sin perder calidad en los servicios ofrecidos. En general, también se podría definir como la capacidad del sistema informático de cambiar su tamaño o configuración para adaptarse a las circunstancias cambiantes.

EV. Las Empresas y Entidades vinculadas (EV) son aquellas que prestan servicios de salud a los asegurados de las Entidades Prestadoras de Salud (EPS). Pueden ser Clínicas, Hospitales, Policlínicos, Centros Médicos, Institutos, Consultorios, Servicios Médicos de Apoyo y Servicios de Atención Domiciliaria. Están inscritas en el Registro correspondiente de la Superintendencia de EPS y están sujetas a su supervisión.

Firewall. Un cortafuegos (o firewall en inglés), es un elemento de hardware o software utilizado en una red de computadoras para controlar las comunicaciones, permitiéndolas o prohibiéndolas según las políticas de red que haya definido la organización responsable de la red.

Grid. La computación distribuida o informática en malla (grid), es un nuevo modelo para resolver problemas de computación masiva utilizando un gran

número de computadoras organizadas en racimos incrustados en una infraestructura de telecomunicaciones distribuida.

Intranet. Una Intranet es una red de computadoras dentro de una red de área local (LAN) privada, empresarial o educativa que proporciona herramientas de Internet. Tiene como función principal proveer lógica de negocios para aplicaciones de captura, informes y consultas con el fin de facilitar la producción de dichos grupos de trabajo; es también un importante medio de difusión de información interna a nivel de grupo de trabajo.

ISP. Un proveedor de servicios de Internet (o ISP por la sigla en idioma inglés de Internet Service Provider) es una empresa dedicada a conectar a Internet a los usuarios o las distintas redes que tengan, y dar el mantenimiento necesario para que el acceso funcione correctamente.

LAN. Una red de área local, o red local, es la interconexión de varios ordenadores y periféricos. (LAN es la abreviatura inglesa de Local Area Network, 'red de área local'). Su extensión esta limitada físicamente a un edificio o a un entorno de hasta 100 metros.

Mainframe. Una computadora central o mainframe es una computadora grande, potente y costosa usada principalmente por una gran compañía para el procesamiento de una gran cantidad de datos; por ejemplo, para el procesamiento de transacciones bancarias.

Marshalling. En ciencias de la computación, la serialización (o marshalling en inglés) consiste en un proceso de codificación de un Objeto (programación orientada a objetos) en un medio de almacenamiento (como puede ser un archivo, o un buffer de memoria) con el fin de transmitirlo a través de una

conexión en red como una serie de bytes o en un formato humanamente más legible como XML.

Modularidad. La modularidad es la capacidad que tiene un sistema de ser estudiado, visto o entendido como la unión de varias partes que interactúan entre sí y que trabajan para alcanzar un objetivo común, realizando cada una de ellas una tarea necesaria para la consecución de dicho objetivo.

Multithreading. Los procesadores con capacidad para multithilo (multithreading en inglés) tienen soporte en hardware para ejecutar eficientemente múltiples hilos de ejecución.

OMG. El Object Management Group u OMG (de sus siglas en inglés Grupo de Gestión de Objetos) es un consorcio dedicado al cuidado y el establecimiento de diversos estándares de tecnologías orientadas a objetos, tales como UML, XMI, CORBA. Es una organización sin ánimo de lucro que promueve el uso de tecnología orientada a objetos mediante guías y especificaciones para las mismas.

PDA. PDA, del inglés Personal Digital Assistant (Asistente Digital Personal), es un computador de mano originalmente diseñado como agenda electrónica (calendario, lista de contactos, bloc de notas y recordatorios) con un sistema de reconocimiento de escritura.

Peer to peer. A grandes rasgos, una red informática entre iguales (en inglés peer-to-peer -que se traduciría de par a par- o de punto a punto, y más conocida como P2P) se refiere a una red que no tiene clientes ni servidores fijos, sino una serie de nodos que se comportan simultáneamente como clientes y como servidores de los demás nodos de la red.

Polimorfismo. En programación orientada a objetos se denomina polimorfismo a la capacidad que tienen los objetos de una clase de responder al mismo mensaje o evento en función de los parámetros utilizados durante su invocación.

Proxy. En el contexto de las redes informáticas, el término proxy hace referencia a un programa o dispositivo que realiza una acción en representación de otro. La finalidad más habitual es la de servidor proxy, que sirve para permitir el acceso a Internet a todos los equipos de una organización cuando sólo se puede disponer de un único equipo conectado, esto es, una única dirección IP.

RMI. RMI (Java Remote Method Invocation) es un mecanismo ofrecido en Java para invocar un método remotamente. Al ser RMI parte estándar del entorno de ejecución Java, usarlo provee un mecanismo simple en una aplicación distribuida que solamente necesita comunicar servidores codificados para Java. Si se requiere comunicarse con otras tecnologías debe usarse CORBA o SOAP en lugar de RMI.

RPC. El RPC (del inglés Remote Procedure Call, Llamada a Procedimiento Remoto) es un protocolo que permite a un programa de ordenador ejecutar código en otra máquina remota sin tener que preocuparse por las comunicaciones entre ambos. El protocolo es un gran avance sobre los sockets usados hasta el momento. De esta manera el programador no tenía que estar pendiente de las comunicaciones, estando éstas encapsuladas dentro de las RPC.

SaaS. El software como servicio (SaaS por sus siglas en inglés: Software as a Service) es un modelo de distribución de software donde la compañía de

software proporciona mantenimiento, operación técnica diaria, y la ayuda para el software proporcionado a su cliente. SaaS es un modelo de entrega de software más que un segmento del mercado; asume que el software está entregado sobre Internet. El software se puede entregar usando este método a cualquier segmento del mercado, desde consumidores caseros hasta corporaciones.

SEPS. Superintendencia de Entidades Prestadoras de Salud, La SEPS es un organismo público descentralizado del Sector Salud, tiene como finalidad el autorizar, regular y supervisar el funcionamiento de las Entidades Prestadoras de Salud, cautelando el uso correcto de los fondos por éstas administrados y el cumplimiento de las normas legales y reglamentarias correspondientes, en resguardo de los derechos de los asegurados.

SOA. La Arquitectura Orientada a Servicios (en inglés Service Oriented Architecture o SOA), es un concepto de arquitectura de software que define la utilización de servicios para dar soporte a los requerimientos de software del usuario.

XDR. XDR (eXternal Data Representation) es un protocolo de presentación de datos, según el Modelo OSI. Permite la transferencia de datos entre máquinas de diferentes arquitecturas y sistemas operativos.

XML. XML, sigla en inglés de Extensible Markup Language («lenguaje de marcas extensible»), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML). Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir

lenguajes para diferentes necesidades. Algunos de estos lenguajes que usan XML para su definición son XHTML, SVG, MathML.

WS-SECURITY. Seguridad en Servicios Web, es un protocolo de comunicaciones que suministra un medio para aplicar seguridad a los Servicios Web. En Abril de 2004 el estándar WS-Security 1.0 fue publicado por Oasis-Open. En 2006 fue publicada la versión 1.1.

ANEXOS

ANEXO 01

Descripción General del Siteds

El Sistema Siteds es un conmutador de mensajes tipo transaccional basado en una arquitectura CLIENTE / SERVIDOR, que permite interconectar las Clínicas o Entidades Vinculadas (CLIENTE), las EPS o Seguros (SERVIDORES) y la SEPS (GATEWAY).

Componentes del SITEDS

Debido a que el sistema está basado en un esquema Cliente-Servidor debemos hablar de los componentes en las diferentes partes que intervienen en el flujo de datos:

- **Cliente**

  Es la estación de trabajo del usuario, el mismo que necesita accesar información de diferentes bases de datos del sistema. Esta compuesto por un software de comunicación y software de aplicación.

- **Servidor EPS (Datos)**

  Es el computador que tiene la base de datos. Está instalado en las EPS, que recibe las transacciones y las responde luego de interactuar con los sistemas informáticos propios de la EPS. Desde el Cliente, se puede accesar directamente a este Servidor, si las políticas de seguridad así lo permiten.

- **Servidor SEPS (Comunicaciones)**

  Es el procesador que se encarga de direccionar los requerimientos de las estaciones hacia los servidores de datos, es decir, administra y enruta los mensajes o transacciones electrónicas y devuelve las respuestas a los primeros, el cual es instalado en el Nodo SEPS.

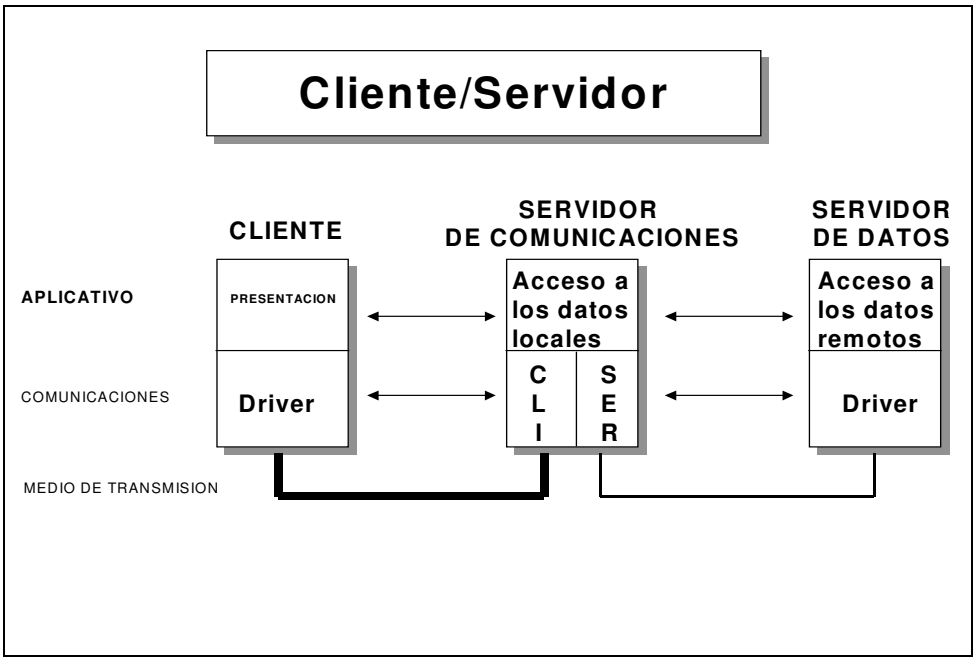

Figura A1. SITEDS Cliente

Funcionalidad General de Componentes

1) Aplicativos y Módulos del Cliente

Cliente Siteds

- Desarrollado en Visual Basic.
- Contiene las pantallas de los Aplicativos de EPS y Seguros.
- Incluye una Base de Datos en Access con los catálogos ha usar, así como una base de datos para almacenar las autorizaciones de EPS y Seguros.
- Use el componente DLLX12 para manejar el formato X12 de las tramas, con el cual maneja una trama propia más simple de intercambio.

DLLX12

- Desarrollado en Visual Basic.
- Maneja el formato X12 de las tramas, tanto para el aplicativo EPS y Seguros.
- Es usado por el Cliente Siteds, el Web Siteds y podría ser usado por cualquier otro aplicativo externo, que desee interactuar con las tramas X12 del Siteds.
- Usa el componente DLLCOM para realizar la comunicación vía TCP/IP con el Servidor de Datos o de Comunicación.

DLLCOM

- Desarrollado en Visual C++.
- Se encarga de establecer la conexión vía el protocolo TCP/IP con los Servidores EPS o SEPS.

2) Módulos del Servidor SEPS

Driver de Comunicación

- Recibe las tramas de consultas de los Aplicativos Clientes, bajo el protocolo TCP/IP.
- Soporta envió de tramas de consultas, así como tramas para transferencia de archivos, las cuales rutea directamente al Servidor EPS.
- Puede recibir más de una consulta a la vez por el mismo puerto.
- Envía la consulta al Ruteador y espera la respuesta la consulta.

- Devuelve la respuesta al Aplicativo Cliente.

Módulos CORE

- Incluye el programa Ruteador, que recibe las consultas de los Driver de Comunicación y se lo pasa al Driver Gateway asociado al aplicativo consultado. Asimismo, recibe las respuestas del Driver Gateway, y se lo deja al Driver de Comunicación que envió la consulta.
- Asimismo, incluye el programa Procesos, que valida que los demás aplicativos estén funcionando y que los recursos estén disponibles.
- Por ultimo, incluye un programa de Auditoria (Log), que registra los datos de las transacciones que pasan por los aplicativos.

Driver Gateway

- Se encarga de recibir la consulta del Ruteador, y conectarse al Servidor EPS, al cual se ha asociado vía configuración.
- Envía la consulta al Servidor EPS y espera la respuesta, la cual luego devuelve al Ruteador.

3) **Módulos EPS**

Driver de Comunicación

- Recibe las tramas de consultas del Servidor SEPS o de los Aplicativos Clientes directamente, bajo el protocolo TCP/IP.
- Soporta envió de tramas de consultas, así como tramas para transferencia de archivos.
- Puede recibir más de una consulta a la vez por el mismo puerto.
- Envía la consulta al Ruteador y espera la respuesta la consulta.
- Devuelve la respuesta al origen de la consulta.

Módulos CORE

- Incluye el programa Ruteador, que recibe las consultas de los Driver de Comunicación y se lo pasa al Driver de Aplicación asociado al aplicativo consultado. Asimismo, recibe las respuestas del Driver de Aplicación, y se lo deja al Driver de Comunicación que envió la consulta.
- Asimismo, incluye el programa Procesos, que valida que los demás aplicativos estén funcionando y que los recursos estén disponibles.
- Por ultimo, incluye un programa de Auditoria (Log), que registra los datos de las transacciones que pasan por los aplicativos.

Driver de Aplicación

- Se encarga de atender la consulta realizada desde el Cliente, ya sea a un archivo plano o una base de datos.
- Los Driver's de Aplicación, pueden ejecutarse varias veces al mismo tiempo, para pos atender la mayor cantidad de consultas en paralelo.

4) Esquema Siteds

Tanto Pacifico como Rímac cuenta con los siguientes componentes para dar soporte al Sistema Siteds:

- Servidor SEPS.
- Servidores EPS.

Con lo cual, el esquema del flujo de transacciones del Siteds, entra las Clínicas (Cliente) y las EPS's es de la siguiente manera:

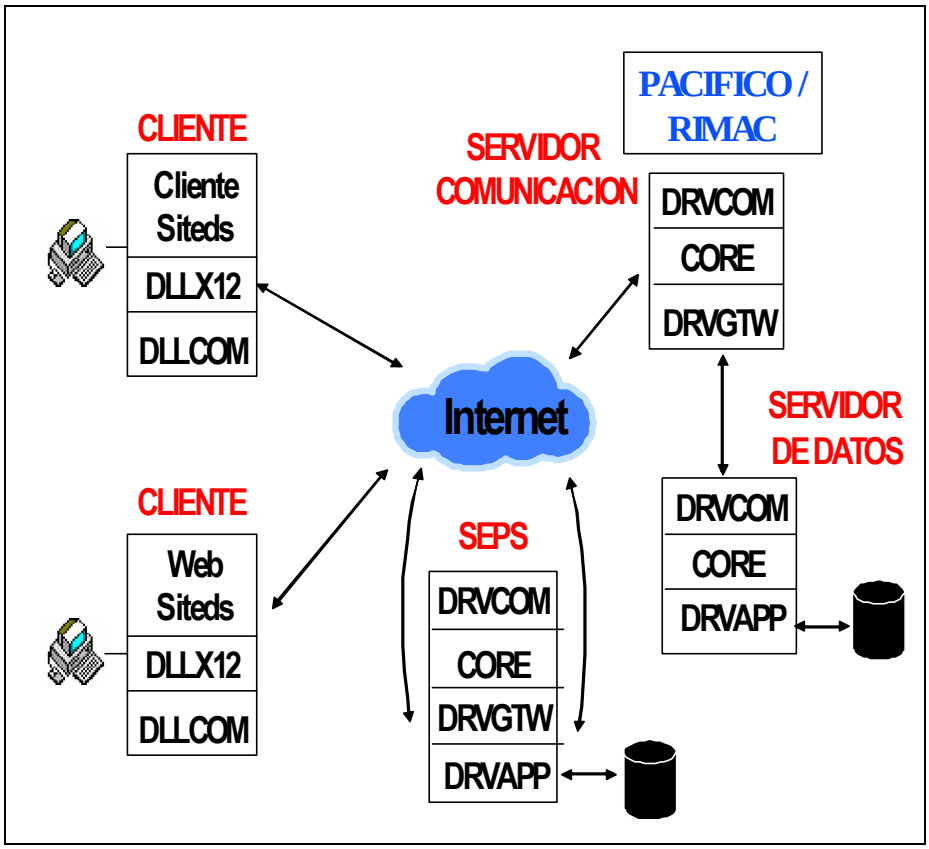

Figura A2. Componentes SITEDS Cliente

El Cliente Siteds, está configurado para acceder por defecto primero al Servidor de la SEPS, y que este lo rutee a los Servidores de EPS, pero en caso haya problemas con la SEPS, el Cliente puede direccionar directamente a los Servidores de las EPS's. En el caso de la Web del Siteds, este solo puede direccionar al Servidor de la SEPS.

ANEXO 02

Tablas del Sistema

Las siguientes tablas contienen los valores necesarios para generar la trama de respuesta de los Servicios Web.

2.1.- Tabla Coberturas

Código de Cobertura	Descripción de la Cobertura
D0	ABORTO / AMENAZA DE ABORTO AMBULATORIO
D1	ABORTO / AMENAZA DE ABORTO HOSPITALARIO
84	ABORTO NO PROVOCADO Y/O AMENAZA
CL	ACCIDENTES
YX	ACCIDENTES DE TRABAJO AMBULATORIO
YZ	ACCIDENTES DE TRABAJO EMERGENCIA
YY	ACCIDENTES DE TRABAJO HOSPITALARIO
XH	ACCIDENTES PERSONALES
Y9	ACUPUNTURA
WY	ADICIONALES
FA	AMB RED
59	AMBULANCIA
EF	AMBULATORIO CON CARTA DE GARANTIA
F2	AMBULATORIO CON RECUPERO
CY	AMBULATORIO EN TÓPICOS TALARA
DB	AMBULATORIO POR TRATAMIENTO
C5	AMBULATORIO POST-OPERATORIO
FK	AMBULATORIO SIN CARTAS DE GARANTIA
F3	AMBULATORIO SIN RECUPERO
CG	APOYO PARCIAL PARA CASOS DE HOSP.
E0	ASISTENCIA DEL VIAJERO
FQ	Asma SP (Sin Protocolo Estándar)
YS	ATEN. DOMICILIARIA LIMA (CARDIOLOGIA, GERIATRIA, DERMA, GASTRO)
DT	ATENC. MEDICA A DOMIC. COBERTURA AMPLIADA
WI	ATENCION DE SALUD PREVENTIVA
YR	ATENCION DOMICILIARIA LIMA (MEDICINA INTERNA, PEDIATRIA)

YT	ATENCION DOMICILIARIA PROVINCIA (MEDICINA INTERNA, PEDIATRIA)
W4	ATENCION MÉDICA A DOMICILIO INTEGRAL
C2	ATENCIONES AMBULATORIAS ESPECIALES
C1	AUDIFONOS
XD	AVION INTERNACIONAL
57	AVION NACIONAL
F1	CAPITACION
D5	CENTRO SALUD
DA	CENTRO SALUD EN OFICINA
FN	CENTRO SALUD EN OFICINA DESCENTRALIZADO
W6	CESAREA
WA	CESAREA/PARTO MULTIPLE/COMPLICACIONES
DM	CHEQUEO GENERAL
ZW	CHEQUEO GINECOLOGICO
F4	CHEQUEO GINECOLOGICO VIP
ZV	CHEQUEO MEDICO GENERAL
F6	CHEQUEO ODONTOLOGICO VIP
F5	CHEQUEO OFTALMOLOGICO VIP
WK	CHEQUEOS VARIOS (VIP)
WJ	CHEQUEOS Y DESPISTAJES
8	CIRUGIA AMBULATORIA
ZB	CIRUGIA CATASTROFICA PRIMER TOPE
ZC	CIRUGIA CATASTROFICA SEGUNDO TOPE
DK	COB. ALTO RIESG-BENEF.VITAL.ONCOL,CARDIOL
30	COBERTURA GENERAL DEL PLAN DE SALUD
D2	COMPLICACIONES EMBARAZO AMBULATORIO
D3	COMPLICACIONES EMBARAZO HOSPITALARIO
W8	COMPLICACIONES EN EMBARAZO Y PUERPERIO
W9	COMPLICACIONES ORGANICAS EMBARAZO
CU	CONDICIONES ESPECIALES
XW	CONSULTA (AMBULATORIO 1 - M.INT/PEDIATR./GINECOLG.)
XX	CONSULTA (AMBULATORIO 2 - OTRAS ESPECIALIDADES)
3	CONSULTA AMBULATORIA
YL	CONSULTA DE REEVALUACION QUIROPRACTICO
ZT	CONSULTA GERIATRICA
C4	CONSULTA MAD ESPECIALISTA
XQ	CONSULTA PSICOLOGICA
DJ	CONSULTA VARIAS
YU	CONSULTAS A DOMICILIO
WD	CONSULTAS PRE/POST NATALES
A4	CONSULTAS PSIQUIATRICAS
C6	CONTINUACION DE EMERGENCIA ACCIDENTAL
DX	CONTINUIDAD DE COBERTURA
WG	CONTROL DE SALUD
68	CONTROL DEL NIÑO SANO - 1 AÑO

65	CONTROL DEL NIÑO SANO
WC	CONTROL POST - NATAL
EA	CONTROL POST HOSPITALARIO
WB	CONTROL PRE - NATAL
XK	COSTO FIJO/TOPICOS
AO	CRISTALES
CT	DELIVERY DE MEDICINAS
WP	DENTAL - NIVEL 1
WQ	DENTAL - NIVEL 2
WR	DENTAL - NIVEL 3
WS	DENTAL - NIVEL 4
WT	DENTAL - NIVEL 5
CC	DEPORTES NO PROFESIONALES
CW	DESAMPARO FAMILIAR - SEGUROS
YV	DESAMPARO SUBITO FAMILIAR
CJ	DESGRAVAMEN DE PENSIONES PENDIENTES DE P
WM	DESPISTAJES (VARIOS)
WL	DESPISTAJES DE CANCER
XG	DIABETES MELLITUS
FS	Diabetes SP (Sin Protocolo Estándar)
FT	Dislipidemia SP (Sin Protocolo Estándar)
FD	EDUCACION PARA LA SALUD
86	EMERGENCIA
E1	EMERGENCIA A DOMICILIO
51	EMERGENCIA ACCIDENTAL
FB	EMERGENCIA ALO RIMAC
52	EMERGENCIA MEDICA
EE	EMERGENCIA OBSTETRICA
EB	EMERGENCIA OBSTÉTRICA AMBULATORIA
Y3	ENCORE ASMA A DOMICILIO
Y2	ENCORE ASMA EN CLINICA
YF	ENCORE DIABETES A DOMICILIO
YE	ENCORE DIABETES EN CLINICA
Y1	ENCORE HTA A DOMICILIO
Y0	ENCORE HTA EN CLINICA
Y5	ENCORE RINITIS A DOMICILIO
Y4	ENCORE RINITIS EN CLINICA
ZP	ENDODONCIA
DI	ENFERMEDADES CONG. DEL RECIEN NACIDO
X4	ENFERMEDADES CONGENITAS
Z4	ENFERMEDADES CONGENITAS AMBULATORIO
Z6	ENFERMEDADES CONGENITAS EMERGENCIA
Z5	ENFERMEDADES CONGENITAS HOSPITALARIO
ZA	ENFERMEDADES CRONICAS ASMA
ZL	ENFERMEDADES CRONICAS DIABETES
ZM	ENFERMEDADES CRONICAS DISLIPIDEMIA
Z9	ENFERMEDADES CRONICAS HTA
ZN	ENFERMEDADES CRONICAS OSTEOPOROSIS
FE	ENFERMEDADES DE CORAZON HOSPITALARIO

ZF	ENFERMEDADES DE TRABAJO AMBULATORIO
ZH	ENFERMEDADES DE TRABAJO EMERGENCIA
ZG	ENFERMEDADES DE TRABAJO HOSPITALARIO
X6	ENFERMEDADES DEL CORAZON
FF	ENFERMEDADES DEL CORAZON AMBULATORIO
CQ	ENFERMEDADES EPIDEMICAS
DL	ENFERMEDADES EPIDERMICAS
YP	EVALUACIÓN MÉDICA INTEGRAL (SALUD JUVENIL)
YB	EVALUACION PSICOLOGICA
WN	EXAMEN MEDICO COMPLEMENTARIO (TITULARES)
YM	EXAMENES ADICIONALES A LA EVALUACION QUIROPRACTICA
YA	EXCIMER LASER
DQ	FARMACIA
ZO	FISIOTERAPIA
ZX	FONOMEDIC PEDIATRIA
CA	GASTOS DE CURACION
XF	HIPERTENSION ARTERIAL (ENCORE)
FR	Hipertensión SP (Sin Protocolo Estándar)
FG	HOMEOPATIA
47	HOSPITALARIO
W2	HOSPITALARIO CLINICO
W1	HOSPITALARIO OBSTETRICO
53	HOSPITALARIO POR CIRUGIA
W0	HOSPITALARIO POR TRATAMIENTO
ZK	HSP CLINICA
ZJ	HSP QUIRURGICA
CE	HUELGAS,CONMOCION CIVIL, DAÑO MALICIOSO,
WW	INDEMNIZACION MENSUAL MUERTE ACCIDENTAL
WV	INDEMNIZACION MENSUAL POR MUERTE NATURAL
XY	INDEMNIZACION POR MUERTE
C9	INVALIDEZ PERMANENTE PARCIAL
C8	INVALIDEZ PERMANENTE TOTAL
XA	LABORATORIO
CO	LIBERACION DE PAGO DE PRIMA
DP	MANEJO DE CASOS
69	MATERNIDAD
YN	MAXISALUD (PROGRAMA DE ENFERMEDADES CRÓNICAS)
D6	MAXISALUD ASMA
D7	MAXISALUD DIABETES
D9	MAXISALUD DISLIPIDEMIA
D8	MAXISALUD HIPERTENSION ARTERIAL
DH	MAXISALUD OBESIDAD
DW	MEDICINA ESTETICA
DR	MEDICINA FISICA Y REHABILITACION
FP	MEDICINA FISICA Y REHABILITACION CON RECUPERO
WZ	MEDICINAS ESPECIALES
YG	MEDICINAS NO ENCORE A DOMICILIO

Y6	MEDICINAS NO ENCORE EN CLINICA
BR	MEDICION DE VISTA
WF	MEDICO EN PLANTA
ZU	MEDICO EN PLANTA ENFERMEDAD CRÓNICO
ZD	MEDICONSULTA
XM	MER/MIR
X3	MONTURA Y CRISTALES
D4	MONTURAS
WU	MUERTE ACCIDENTAL
EC	NUTRICIÓN 1RA CONSULTA
ED	NUTRICIÓN CONSULTA SEGUIMIENTO
WO	ODONTOLOGIA
X1	OFTALMOLOGIA
Y7	OFTALMOLOGIA AMBULATORIA
Y8	OFTALMOLOGIA HOSPITALARIA
FL	OFTALMOLOGICO AMBULATORIO CON CARTA DE GARANTIA
DC	OFTALMOLOGICO AMBULATORIO POR CIRUGIA
DE	OFTALMOLOGICO AMBULATORIO POR TRATAMIENTO
FM	OFTALMOLOGICO AMBULATORIO SIN CARTA DE GARANTIA
DF	OFTALMOLOGICO HOSPITALARIO POR CIRUGIA
DG	OFTALMOLOGICO HOSPITALARIO POR TRATAMIENTO
X7	ONCOLOGIA
FC	ONCOLOGICO (CAPITACION)
BJ	ONCOLOGICO AMBULATORIO
W3	ONCOLOGICO HOSPITALARIO
YQ	ORIENTACIÓN PSICOLOGICA (SALUD JUVENIL)
ZQ	ORTODONCIA
X5	OTRAS ENFERMEDADES
CI	PAGO DE PENSION ESTUD. INVALIDEZ PERMANE
CH	PAGO DE PENSION ESTUD. MUERTE ACCIDENTAL
W7	PARTO MULTIPLE
W5	PARTO NORMAL
CB	PASAJERO Y/O CONDUCTOR DE CUALQUIER
CN	PERFIL OBSTETRICO PRE NATAL (1ER TRIMES)
82	PLANIFICACION FAMILIAR
CX	PRE/POST NATAL - SEGUROS
YO	PREAFILIACION (CHEQUEO MEDICO)
WH	PREVENCION
23	PREVENCION DENTAL
X2	PREVENCION OFTALMOLOGICA
DO	PREVENCION SCTR
XT	PREVENTIVO PROMOCIONAL
X9	PROBLEMA PSIQUIATRICO DE ORIGEN ORGANICO
CR	PROCEDIMIENTOS VASCULARES
DN	PROGRAMA DE CONTROL DE ENFERMEDADES
DZ	PROGRAMA DE NUTRICION
XE	PROGRAMA ENCORE
XR	PROINVESTIGACIÓN

YW	PROINVESTIGACIÓN DENTALES
X0	PROINVESTIGACIÓN QUIRURGICAS
CV	PSICOLOGIA - SEGUROS
WE	PSICOPROFILAXIS
YC	PSICOTERAPIA EN SESION GRUPAL
YD	PSICOTERAPIA EN SESION INDIVIDUAL
YI	QUIROPRACTICO EN ADULTO
YH	QUIROPRACTICO EN NIÑO
FH	RED PRIVADA RIMAC
DU	RENOVACION DE RECETAS
Z8	REPATRIACION DE RESTOS
ZE	REVALUACIONES MAD
CK	RIESGO QUIRURGICO
F9	RIESGO QUIRURGICO PARA HOSPITALIZACION
FU	Rinitis SP (Sin Protocolo Estándar)
XB	SALUD DE LA MUJER
X8	SALUD MENTAL
ZS	SALUD MENTAL AMBULATORIO
ZR	SALUD MENTAL HOSPITALIZACION
CM	SCREENING GENETICO PRE NATAL
XJ	SEGUNDA CAPA
XU	SEGUNDA OPINION EN EL EXTRANJERO
FJ	SEGUNDA OPINION NACIONAL
20	SEGUNDA OPINION QUIRURGICA
CS	SEGURO A BORDO
C7	SEGURO DE VIDA
XS	SEGURO PRIMAS POR MUERTE TITULAR
WX	SEPELIO
E5	SEPELIO PLUS
E7	SEPELIO PLUS (P)
E8	SEPELIO PREFERENCIAL (P)
E2	SEPELIO PREMIUM 1
E3	SEPELIO PREMIUM 2
E4	SEPELIO PREMIUM 3
E6	SEPELIO VIP (P)
YK	SESION DE TERAPIA FISICA QUIROPRACTICO
YJ	SESION QUIROPRACTICO
Z0	SUBSIDIO HOSPIT EN ESSALUD CAPA COMPLEJA
CD	TERCERA OPINION MEDICA
CF	TERREMOTOS Y/O FENOMENOS NATURALES
ZI	TOPICO
CP	TRANSPLANTE DE ORGANOS
XN	TRANSPORTE AEREO - EVACUACION EN AMBULANCIA
56	TRANSPORTE POR EVACUACION
Z7	TRASLADO DE RESTOS
XI	TRASLADOS
DY	TRATAMIENTO DE INFERTILIDAD
A6	TRATAMIENTO PSIQUIAT. ALCOH. DROGADICC
DV	TRATAMIENTO QUIROPRACTICO
C3	TRATAMIENTOS ALTERNATIVOS
A8	TRATMTO.PSIQ/ALCOH/DROGAD. AMBULATORIO
A7	TRATMTO.PSIQ/ALCOH/DROGAD.HOSPITALARIO

XL	URGENCIA ASISTIDA
XC	VACUNAS E INMUNIZACIONES
Z1	VIH "SIDA" AMBULATORIO
Z3	VIH "SIDA" EMERGENCIA
Z2	VIH "SIDA" HOSPITALARIO
DS	VISITA POR DESCANSO MEDICO
79	ZAPATOS ORTOPEDICOS O PLANTILLAS

2.2.- Tabla de Servicios

Código Servicio	Descripción del Servicio
99	CANTIDAD USADA
CA	CUBIERTA - ACTUAL
CE	CUBIERTA - ESTIMADA
DB	UNIDADES DEDUCIBLES DE SANGRE
DY	DIAS
HS	HORAS
LA	RESERVA DE TIEMPO DE VIDA - ACTUAL
LE	RESERVA DE TIEMPO DE VIDA - ESTIMADA
MN	MES
P6	NUMERO DE SERVICIOS O PROCEDIMIENTOS
QA	CANTIDAD APROBADA
S7	EDAD, MAXIMO VALOR
S8	EDAD, VALOR MINIMO
VS	ATENCION
YY	ANUAL
Z3	DIA DE CUARTO
ZU	CONSULTA
ZV	PIEZA DENTAL TRATADA
ZW	AMALGAMA COMPUESTA
ZX	AMBULANCIA
ZY	ECOGRAFIA
ZZ	EXIMER LASER

2.3.- Tabla Tipo de Afiliación

Código Tipo Afiliación	Descripción de la Afiliación
1	REGULAR
2	POTESTATIVA
3	SCTR

2.4.- Tabal de Parentesco con el Titular

Código de Parentesco	Descripción del Parentesco
1	CONYUGE
3	PADRE O MADRE
8	PRIMO
18	TITULAR
19	HIJO O HIJA
21	DESCONOCIDO

22	HIJO O HIJA INCAPACITADO
32	MADRE
33	PADRE
45	VIUDO
61	TIA
62	HERMANO
74	HIJA
95	NIETO
96	NIETA
97	ABUELO
98	ABUELA
A6	ESPOSO
B1	SOBRINO
B2	SOBRINA
B7	HERMANA
C1	HIJO
C2	YERNO
C3	SUEGRO
D3	TIO
D4	ESPOSA

2.5.- Tabla Tipo Producto

Código Tipo producto	Descripción del Tipo de producto
R	SCTR
S	SALUD
A	AMBOS

2.6.- Tabla Tipo de Estado del Asegurado

Código del Estado	Descripción
1	COBERTURA ACTIVA
6	INACTIVA
7	SUSPENDIDO
I	NO CUBIERTO

2.7.- Tabla de Medio de Comunicación

Código	Descripción
EM	CORREO ELECTRONICO
EX	ANEXO TELEFONICO
FX	FAX
HP	TELEFONO DEL HOGAR
TE	TELEFONO
WP	TELEFONO DE TRABAJO

2.8.- Tabla de Tipo de Autorización

Código	Descripción
1	AUTORIZACIÓN DE CONSULTA DE ELEGIBILIDAD
2	AUTORIZACIÓN DE CARTAS DE GARANTÍA

2.9.- Tabla de Indicador de Procedimientos

Código	Descripción
1	SIN PROCEDIMIENTOS
2	TIENE PROCEDIMIENTOS

ANEXO 03

Estructura Estándar de la Trama de Respuesta

En la trama cada campo será limitado por un asterisco '*' y cada fila del listado de coberturas y preexistencias será limitado por un '~'.

3.1.- Trama respuesta Consulta por Nombre EPS

Campo	Longitud	Tipo	Valor
Apellido Paterno	30	Texto	
Nombres	30	Texto	
Apellido Materno	30	Texto	
Código Afilado	10	Texto	
Estado	1	Texto	Anexo Nº 2 , Tabla 2.6
Tipo Producto	1	Texto	
Fecha Nacimiento	8	Texto	YYYYMMDD
Sexo	1	Texto	F: Femenino, M: Masculino

3.2. – Trama respuesta Consulta por Código de Asegurado

LA Trama generada a partir de la consulta por código de Asegurada, es una concatenación de las tramas 3.2.1, 3.2.2 y 3.2.3.

3.2.1.- Trama de Datos del Asegurado

Campo	Longitud	Tipo	Valor
Numero Solicitud	14	Texto	
Apellido Paterno Paciente	30	Texto	
Apellido Materno Paciente	30	Texto	
Nombres Paciente	30	Texto	
Fecha Nacimiento	8	Texto	YYYYMMDD
Sexo	1	Texto	F: Femenino, M: Masculino
Parentesco	4	Texto	Anexo Nº 2 , Tabla 2.4
Nro de DNI	10	Texto	
Nro Carné Afiliado	10	Texto	
Estado	1	Texto	Anexo Nº 2 , Tabla 2.6

Plan de Salud	5	Texto	
Tipo Afiliación	1	Texto	Anexo Nº 2 , Tabla 2.3
SCR	2	Texto	X
Fecha Inicio Vigencia	8	Texto	YYYYMMDD
Fecha Fin Vigencia	8	Texto	YYYYMMDD
Razón Social Contratante	40	Texto	
RUC Contratante	11	Texto	
Apellido Paterno Titular	30	Texto	
Apellido Materno Titular	30	Texto	
Nombres Titular	30	Texto	
Nro Contrato	11	Texto	
Moneda Deducible	3	Texto	PEN = Soles USD = Dólares
Fecha Inclusión Titular	8	Texto	YYYYMMDD
Observaciones	60	Texto	
Condiciones Especiales	60	Texto	
Flag Foto Asegurado	1	Texto	
Fecha de Foto	10	Texto	

3.2.2.- Trama de Coberturas o Beneficios

Campo	Longitud	Tipo	Valor
Tipo Cobertura	3	Texto	Anexo Nº 2 , Tabla 2.1
Indicador Procedimientos	1	Texto	Anexo Nº 2 , Tabla 2.9
Plan	5	Texto	
Fecha Fin de Carencia	8	Texto	YYYYMMDD
Servicio	2	Texto	Anexo Nº 2 , Tabla 2.2
Cantidad	5	Texto	
Parentesco Autorización	10	Texto	Anexo Nº 2 , Tabla 2.4
Sexo Autorización	1	Texto	F: Femenino, M: Masculino
Observaciones	30	Texto	
Copago Fijo (Deducible)	9	Texto	
Copago Variable (Coaseguro)	6	Texto	
Observaciones	30	Texto	

3.2.3.- Trama de Pre existencias

Campo	Longitud	Tipo	Valor
Código Afilado	10	Texto	
Diagnóstico (CIE10)	5	Texto	Tabla CIE10 Estandarizada SEPS http://www.seps.gob.pe/
Observación	50	Texto	

3.3.- Trama respuesta Datos Adicionales

Campo	Longitud	Tipo	Valor
Dirección 1	50	Texto	
Dirección 2	50	Texto	
Ubigeo	6	Texto	http://www.seps.gob.pe/estandares
Nombre del Contacto	50	Texto	
Tipo Teléfono 1	2	Texto	Anexo Nº 2 , Tabla 2.7
Teléfono 1	7	Texto	
Tipo Teléfono 2	2	Texto	Anexo Nº 2 , Tabla 2.7
Teléfono 2	7	Texto	
Tipo Teléfono 3	2	Texto	Anexo Nº 2 , Tabla 2.7
Teléfono 3	7	Texto	

3.3.- Trama respuesta de Observaciones del Asegurado

Campo	Longitud	Tipo
Observaciones	300	Texto
Condiciones Especiales	300	Texto

3.4.- Trama respuesta Autorización de la atención

Campo	Longitud	Tipo	Valor
Nº de autorización generada	10	Texto	
Tipo de Autorización	1	Texto	Anexo Nº 2 , Tabla 2.8
Observación de autorización	50	Texto	

ANEXO 04

Descripción de los Servicios Web

4.1. - Scwsseps.asmx?WSDL (SEPS)

```
<?xml version="1.0" encoding="utf-8" ?>
- <wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
    xmlns:tns="http://www.seps.gob.pe/cwsseps/scwsseps"
    xmlns:s="http://www.w3.org/2001/XMLSchema"
    xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
    xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
    targetNamespace="http://www.seps.gob.pe/cwsseps/scwsseps"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
- <wsdl:types>
- <s:schema elementFormDefault="qualified"
    targetNamespace="http://www.seps.gob.pe/cwsseps/scwsseps">
- <s:element name="GetConsutaxNombre">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="Entidad_Codigo" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="EVinculada_Codigo" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="EVinculada_Ruc" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="PaternoAsegurado" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="ApMaternoAsegurado" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="NombreAsegurado" type="s:string" />
    </s:sequence>
    </s:complexType>
    </s:element>
- <s:element name="GetConsutaxNombreResponse">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="GetConsutaxNombreResult" type="s:string"
    />
    </s:sequence>
    </s:complexType>
    </s:element>
  <s:element name="AuthenticationHeader" type="tns:AuthenticationHeader" />
- <s:complexType name="AuthenticationHeader">
- <s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="UserName" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="Password" type="s:string" />
    </s:sequence>
  <s:anyAttribute />
```

```
    </s:complexType>
- <s:element name="GetConsultaxCodigo">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="Entidad_Codigo" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="EVinculada_Codigo" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="EVinculada_Ruc" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="Asegurado_Codigo" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="Asegurado_Plan" type="s:string" />
    </s:sequence>
    </s:complexType>
    </s:element>
- <s:element name="GetConsultaxCodigoResponse">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="GetConsultaxCodigoResult" type="s:string" />
    </s:sequence>
    </s:complexType>
    </s:element>
- <s:element name="GetDatosOtros">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="Entidad_Codigo" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="EVinculada_Codigo" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="EVinculada_Ruc" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="Asegurado_Codigo" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="Tipo" type="s:string" />
    </s:sequence>
    </s:complexType>
    </s:element>
- <s:element name="GetDatosOtrosResponse">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="GetDatosOtrosResult" type="s:string" />
    </s:sequence>
    </s:complexType>
    </s:element>
- <s:element name="GetNumAutorizacion">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="Entidad_Codigo" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="EVinculada_Codigo" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="EVinculada_Ruc" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="Asegurado_Codigo" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="Estado" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="Moneda" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="PlandeSalud" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="Observaciones" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="Condiciones" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="TipoCobertura" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="Indicador" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="Plan" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="FechaFinCarencia" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="Servicio" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="Cantidad" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="ParentescoAuto" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="SexoAuto" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="CopagoFijo" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="CopagoVariable" type="s:string" />
    </s:sequence>
```

```
      </s:complexType>
      </s:element>
- <s:element name="GetNumAutorizacionResponse">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="GetNumAutorizacionResult" type="s:string" />
      </s:sequence>
      </s:complexType>
      </s:element>
- <s:element name="GetFoto">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="Entidad_Codigo" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="EVinculada_Codigo" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="EVinculada_Ruc" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="Asegurado_Codigo" type="s:string" />
      </s:sequence>
      </s:complexType>
      </s:element>
- <s:element name="GetFotoResponse">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="1" maxOccurs="1" name="GetFotoResult" type="s:unsignedByte" />
      </s:sequence>
      </s:complexType>
      </s:element>
      </s:schema>
      </wsdl:types>
- <wsdl:message name="GetConsutaxNombreSoapIn">
  <wsdl:part name="parameters" element="tns:GetConsutaxNombre" />
      </wsdl:message>
- <wsdl:message name="GetConsutaxNombreSoapOut">
  <wsdl:part name="parameters" element="tns:GetConsutaxNombreResponse" />
      </wsdl:message>
- <wsdl:message name="GetConsutaxNombreAuthenticationHeader">
  <wsdl:part name="AuthenticationHeader" element="tns:AuthenticationHeader" />
      </wsdl:message>
- <wsdl:message name="GetConsultaxCodigoSoapIn">
  <wsdl:part name="parameters" element="tns:GetConsultaxCodigo" />
      </wsdl:message>
- <wsdl:message name="GetConsultaxCodigoSoapOut">
  <wsdl:part name="parameters" element="tns:GetConsultaxCodigoResponse" />
      </wsdl:message>
- <wsdl:message name="GetConsultaxCodigoAuthenticationHeader">
  <wsdl:part name="AuthenticationHeader" element="tns:AuthenticationHeader" />
      </wsdl:message>
- <wsdl:message name="GetDatosOtrosSoapIn">
  <wsdl:part name="parameters" element="tns:GetDatosOtros" />
      </wsdl:message>
- <wsdl:message name="GetDatosOtrosSoapOut">
  <wsdl:part name="parameters" element="tns:GetDatosOtrosResponse" />
      </wsdl:message>
- <wsdl:message name="GetDatosOtrosAuthenticationHeader">
  <wsdl:part name="AuthenticationHeader" element="tns:AuthenticationHeader" />
      </wsdl:message>
- <wsdl:message name="GetNumAutorizacionSoapIn">
  <wsdl:part name="parameters" element="tns:GetNumAutorizacion" />
      </wsdl:message>
- <wsdl:message name="GetNumAutorizacionSoapOut">
  <wsdl:part name="parameters" element="tns:GetNumAutorizacionResponse" />
```

```
    </wsdl:message>
- <wsdl:message name="GetNumAutorizacionAuthenticationHeader">
  <wsdl:part name="AuthenticationHeader" element="tns:AuthenticationHeader" />
    </wsdl:message>
- <wsdl:message name="GetFotoSoapIn">
  <wsdl:part name="parameters" element="tns:GetFoto" />
    </wsdl:message>
- <wsdl:message name="GetFotoSoapOut">
  <wsdl:part name="parameters" element="tns:GetFotoResponse" />
    </wsdl:message>
- <wsdl:message name="GetFotoAuthenticationHeader">
  <wsdl:part name="AuthenticationHeader" element="tns:AuthenticationHeader" />
    </wsdl:message>
- <wsdl:portType name="scwssepsSoap">
- <wsdl:operation name="GetConsutaxNombre">
  <wsdl:input message="tns:GetConsutaxNombreSoapIn" />
  <wsdl:output message="tns:GetConsutaxNombreSoapOut" />
    </wsdl:operation>
- <wsdl:operation name="GetConsultaxCodigo">
  <wsdl:input message="tns:GetConsultaxCodigoSoapIn" />
  <wsdl:output message="tns:GetConsultaxCodigoSoapOut" />
    </wsdl:operation>
- <wsdl:operation name="GetDatosOtros">
  <wsdl:input message="tns:GetDatosOtrosSoapIn" />
  <wsdl:output message="tns:GetDatosOtrosSoapOut" />
    </wsdl:operation>
- <wsdl:operation name="GetNumAutorizacion">
  <wsdl:input message="tns:GetNumAutorizacionSoapIn" />
  <wsdl:output message="tns:GetNumAutorizacionSoapOut" />
    </wsdl:operation>
- <wsdl:operation name="GetFoto">
  <wsdl:input message="tns:GetFotoSoapIn" />
  <wsdl:output message="tns:GetFotoSoapOut" />
    </wsdl:operation>
    </wsdl:portType>
- <wsdl:binding name="scwssepsSoap" type="tns:scwssepsSoap">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
- <wsdl:operation name="GetConsutaxNombre">
  <soap:operation
      soapAction="http://www.seps.gob.pe/cwsseps/scwsseps/GetConsutaxNombre"
      style="document" />
- <wsdl:input>
  <soap:body use="literal" />
  <soap:header message="tns:GetConsutaxNombreAuthenticationHeader"
      part="AuthenticationHeader" use="literal" />
    </wsdl:input>
- <wsdl:output>
  <soap:body use="literal" />
  <soap:header message="tns:GetConsutaxNombreAuthenticationHeader"
      part="AuthenticationHeader" use="literal" />
    </wsdl:output>
    </wsdl:operation>
- <wsdl:operation name="GetConsultaxCodigo">
  <soap:operation
      soapAction="http://www.seps.gob.pe/cwsseps/scwsseps/GetConsultaxCodigo"
      style="document" />
- <wsdl:input>
  <soap:body use="literal" />
  <soap:header message="tns:GetConsultaxCodigoAuthenticationHeader"
      part="AuthenticationHeader" use="literal" />
```

```
    </wsdl:input>
- <wsdl:output>
  <soap:body use="literal" />
  <soap:header message="tns:GetConsultaxCodigoAuthenticationHeader"
      part="AuthenticationHeader" use="literal" />
    </wsdl:output>
    </wsdl:operation>
- <wsdl:operation name="GetDatosOtros">
  <soap:operation soapAction="http://www.seps.gob.pe/cwsseps/scwsseps/GetDatosOtros"
      style="document" />
- <wsdl:input>
  <soap:body use="literal" />
  <soap:header message="tns:GetDatosOtrosAuthenticationHeader"
      part="AuthenticationHeader" use="literal" />
    </wsdl:input>
- <wsdl:output>
  <soap:body use="literal" />
  <soap:header message="tns:GetDatosOtrosAuthenticationHeader"
      part="AuthenticationHeader" use="literal" />
    </wsdl:output>
    </wsdl:operation>
- <wsdl:operation name="GetNumAutorizacion">
  <soap:operation
      soapAction="http://www.seps.gob.pe/cwsseps/scwsseps/GetNumAutorizacion"
      style="document" />
- <wsdl:input>
  <soap:body use="literal" />
  <soap:header message="tns:GetNumAutorizacionAuthenticationHeader"
      part="AuthenticationHeader" use="literal" />
    </wsdl:input>
- <wsdl:output>
  <soap:body use="literal" />
  <soap:header message="tns:GetNumAutorizacionAuthenticationHeader"
      part="AuthenticationHeader" use="literal" />
    </wsdl:output>
    </wsdl:operation>
- <wsdl:operation name="GetFoto">
  <soap:operation soapAction="http://www.seps.gob.pe/cwsseps/scwsseps/GetFoto"
      style="document" />
- <wsdl:input>
  <soap:body use="literal" />
  <soap:header message="tns:GetFotoAuthenticationHeader" part="AuthenticationHeader"
      use="literal" />
    </wsdl:input>
- <wsdl:output>
  <soap:body use="literal" />
  <soap:header message="tns:GetFotoAuthenticationHeader" part="AuthenticationHeader"
      use="literal" />
    </wsdl:output>
    </wsdl:operation>
    </wsdl:binding>
- <wsdl:binding name="scwssepsSoap12" type="tns:scwssepsSoap">
  <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />
- <wsdl:operation name="GetConsutaxNombre">
  <soap12:operation
      soapAction="http://www.seps.gob.pe/cwsseps/scwsseps/GetConsutaxNombre"
      style="document" />
- <wsdl:input>
  <soap12:body use="literal" />
```

```
  <soap12:header message="tns:GetConsutaxNombreAuthenticationHeader"
      part="AuthenticationHeader" use="literal" />
    </wsdl:input>
- <wsdl:output>
  <soap12:body use="literal" />
  <soap12:header message="tns:GetConsutaxNombreAuthenticationHeader"
      part="AuthenticationHeader" use="literal" />
    </wsdl:output>
    </wsdl:operation>
- <wsdl:operation name="GetConsultaxCodigo">
  <soap12:operation
      soapAction="http://www.seps.gob.pe/cwsseps/scwsseps/GetConsultaxCodigo"
      style="document" />
- <wsdl:input>
  <soap12:body use="literal" />
  <soap12:header message="tns:GetConsultaxCodigoAuthenticationHeader"
      part="AuthenticationHeader" use="literal" />
    </wsdl:input>
- <wsdl:output>
  <soap12:body use="literal" />
  <soap12:header message="tns:GetConsultaxCodigoAuthenticationHeader"
      part="AuthenticationHeader" use="literal" />
    </wsdl:output>
    </wsdl:operation>
- <wsdl:operation name="GetDatosOtros">
  <soap12:operation soapAction="http://www.seps.gob.pe/cwsseps/scwsseps/GetDatosOtros"
      style="document" />
- <wsdl:input>
  <soap12:body use="literal" />
  <soap12:header message="tns:GetDatosOtrosAuthenticationHeader"
      part="AuthenticationHeader" use="literal" />
    </wsdl:input>
- <wsdl:output>
  <soap12:body use="literal" />
  <soap12:header message="tns:GetDatosOtrosAuthenticationHeader"
      part="AuthenticationHeader" use="literal" />
    </wsdl:output>
    </wsdl:operation>
- <wsdl:operation name="GetNumAutorizacion">
  <soap12:operation
      soapAction="http://www.seps.gob.pe/cwsseps/scwsseps/GetNumAutorizacion"
      style="document" />
- <wsdl:input>
  <soap12:body use="literal" />
  <soap12:header message="tns:GetNumAutorizacionAuthenticationHeader"
      part="AuthenticationHeader" use="literal" />
    </wsdl:input>
- <wsdl:output>
  <soap12:body use="literal" />
  <soap12:header message="tns:GetNumAutorizacionAuthenticationHeader"
      part="AuthenticationHeader" use="literal" />
    </wsdl:output>
    </wsdl:operation>
- <wsdl:operation name="GetFoto">
  <soap12:operation soapAction="http://www.seps.gob.pe/cwsseps/scwsseps/GetFoto"
      style="document" />
- <wsdl:input>
  <soap12:body use="literal" />
  <soap12:header message="tns:GetFotoAuthenticationHeader" part="AuthenticationHeader"
      use="literal" />
```

```
    </wsdl:input>
- <wsdl:output>
  <soap12:body use="literal" />
  <soap12:header message="tns:GetFotoAuthenticationHeader" part="AuthenticationHeader"
      use="literal" />
    </wsdl:output>
    </wsdl:operation>
    </wsdl:binding>
- <wsdl:service name="scwsseps">
- <wsdl:port name="scwssepsSoap" binding="tns:scwssepsSoap">
  <soap:address location="http://localhost:4802/cwsseps/scwsseps.asmx" />
    </wsdl:port>
- <wsdl:port name="scwssepsSoap12" binding="tns:scwssepsSoap12">
  <soap12:address location="http://localhost:4802/cwsseps/scwsseps.asmx" />
    </wsdl:port>
    </wsdl:service>
    </wsdl:definitions>
```

4.1. - wsmapfre.asmx?WSDL (EPS)

```
    <?xml version="1.0" encoding="utf-8" ?>
- <wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
      xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
      xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
      xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
      xmlns:tns="http://tempuri.org/WSMapfre/Service"
      xmlns:s="http://www.w3.org/2001/XMLSchema"
      xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
      xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
      targetNamespace="http://tempuri.org/WSMapfre/Service"
      xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
- <wsdl:types>
- <s:schema elementFormDefault="qualified"
      targetNamespace="http://tempuri.org/WSMapfre/Service">
- <s:element name="GetConsutaxNombre">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="Entidad_Codigo" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="EVinculada_Codigo" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="EVinculada_Ruc" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="PaternoAsegurado" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="ApMaternoAsegurado" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="NombreAsegurado" type="s:string" />
    </s:sequence>
    </s:complexType>
    </s:element>
- <s:element name="GetConsutaxNombreResponse">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="GetConsutaxNombreResult" type="s:string"
      />
    </s:sequence>
    </s:complexType>
    </s:element>
- <s:element name="GetConsultaxCodigo">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="Entidad_Codigo" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="EVinculada_Codigo" type="s:string" />
```

```
<s:element minOccurs="0" maxOccurs="1" name="EVinculada_Ruc" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="Asegurado_Codigo" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="Asegurado_Plan" type="s:string" />
</s:sequence>
</s:complexType>
</s:element>
- <s:element name="GetConsultaxCodigoResponse">
- <s:complexType>
- <s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="GetConsultaxCodigoResult" type="s:string" />
</s:sequence>
</s:complexType>
</s:element>
- <s:element name="GetDatosOtros">
- <s:complexType>
- <s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="Entidad_Codigo" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="EVinculada_Codigo" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="EVinculada_Ruc" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="Asegurado_Codigo" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="Tipo" type="s:string" />
</s:sequence>
</s:complexType>
</s:element>
- <s:element name="GetDatosOtrosResponse">
- <s:complexType>
- <s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="GetDatosOtrosResult" type="s:string" />
</s:sequence>
</s:complexType>
</s:element>
- <s:element name="GetNumAutorizacion">
- <s:complexType>
- <s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="Entidad_Codigo" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="EVinculada_Codigo" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="EVinculada_Ruc" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="Asegurado_Codigo" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="Estado" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="Moneda" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="PlandeSalud" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="Observaciones" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="Condiciones" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="TipoCobertura" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="Indicador" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="Plan" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="FechaFinCarencia" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="Servicio" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="Cantidad" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="ParentescoAuto" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="SexoAuto" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="CopagoFijo" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="CopagoVariable" type="s:string" />
</s:sequence>
</s:complexType>
</s:element>
- <s:element name="GetNumAutorizacionResponse">
- <s:complexType>
- <s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="GetNumAutorizacionResult" type="s:string" />
```

```
      </s:sequence>
      </s:complexType>
      </s:element>
- <s:element name="GetFoto">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="Entidad_Codigo" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="EVinculada_Codigo" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="EVinculada_Ruc" type="s:string" />
  <s:element minOccurs="0" maxOccurs="1" name="Asegurado_Codigo" type="s:string" />
      </s:sequence>
      </s:complexType>
      </s:element>
- <s:element name="GetFotoResponse">
- <s:complexType>
- <s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="GetFotoResult" type="s:string" />
      </s:sequence>
      </s:complexType>
      </s:element>
      </s:schema>
      </wsdl:types>
- <wsdl:message name="GetConsutaxNombreSoapIn">
  <wsdl:part name="parameters" element="tns:GetConsutaxNombre" />
      </wsdl:message>
- <wsdl:message name="GetConsutaxNombreSoapOut">
  <wsdl:part name="parameters" element="tns:GetConsutaxNombreResponse" />
      </wsdl:message>
- <wsdl:message name="GetConsultaxCodigoSoapIn">
  <wsdl:part name="parameters" element="tns:GetConsultaxCodigo" />
      </wsdl:message>
- <wsdl:message name="GetConsultaxCodigoSoapOut">
  <wsdl:part name="parameters" element="tns:GetConsultaxCodigoResponse" />
      </wsdl:message>
- <wsdl:message name="GetDatosOtrosSoapIn">
  <wsdl:part name="parameters" element="tns:GetDatosOtros" />
      </wsdl:message>
- <wsdl:message name="GetDatosOtrosSoapOut">
  <wsdl:part name="parameters" element="tns:GetDatosOtrosResponse" />
      </wsdl:message>
- <wsdl:message name="GetNumAutorizacionSoapIn">
  <wsdl:part name="parameters" element="tns:GetNumAutorizacion" />
      </wsdl:message>
- <wsdl:message name="GetNumAutorizacionSoapOut">
  <wsdl:part name="parameters" element="tns:GetNumAutorizacionResponse" />
      </wsdl:message>
- <wsdl:message name="GetFotoSoapIn">
  <wsdl:part name="parameters" element="tns:GetFoto" />
      </wsdl:message>
- <wsdl:message name="GetFotoSoapOut">
  <wsdl:part name="parameters" element="tns:GetFotoResponse" />
      </wsdl:message>
- <wsdl:message name="CreateDimeDocSoapIn">
  <wsdl:part name="parameters" element="tns:CreateDimeDoc" />
      </wsdl:message>
- <wsdl:message name="CreateDimeDocSoapOut">
  <wsdl:part name="parameters" element="tns:CreateDimeDocResponse" />
      </wsdl:message>
- <wsdl:portType name="ServiceSoap">
- <wsdl:operation name="GetConsutaxNombre">
```

```
  <wsdl:input message="tns:GetConsutaxNombreSoapIn" />
  <wsdl:output message="tns:GetConsutaxNombreSoapOut" />
    </wsdl:operation>
- <wsdl:operation name="GetConsultaxCodigo">
  <wsdl:input message="tns:GetConsultaxCodigoSoapIn" />
  <wsdl:output message="tns:GetConsultaxCodigoSoapOut" />
    </wsdl:operation>
- <wsdl:operation name="GetDatosOtros">
  <wsdl:input message="tns:GetDatosOtrosSoapIn" />
  <wsdl:output message="tns:GetDatosOtrosSoapOut" />
    </wsdl:operation>
- <wsdl:operation name="GetNumAutorizacion">
  <wsdl:input message="tns:GetNumAutorizacionSoapIn" />
  <wsdl:output message="tns:GetNumAutorizacionSoapOut" />
    </wsdl:operation>
- <wsdl:operation name="GetFoto">
  <wsdl:input message="tns:GetFotoSoapIn" />
  <wsdl:output message="tns:GetFotoSoapOut" />
    </wsdl:operation>
    </wsdl:portType>
- <wsdl:binding name="ServiceSoap" type="tns:ServiceSoap">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
- <wsdl:operation name="GetConsutaxNombre">
  <soap:operation soapAction="http://tempuri.org/WSMapfre/Service/GetConsutaxNombre"
      style="document" />
- <wsdl:input>
  <soap:body use="literal" />
    </wsdl:input>
- <wsdl:output>
  <soap:body use="literal" />
    </wsdl:output>
    </wsdl:operation>
- <wsdl:operation name="GetConsultaxCodigo">
  <soap:operation soapAction="http://tempuri.org/WSMapfre/Service/GetConsultaxCodigo"
      style="document" />
- <wsdl:input>
  <soap:body use="literal" />
    </wsdl:input>
- <wsdl:output>
  <soap:body use="literal" />
    </wsdl:output>
    </wsdl:operation>
- <wsdl:operation name="GetDatosOtros">
  <soap:operation soapAction="http://tempuri.org/WSMapfre/Service/GetDatosOtros"
      style="document" />
- <wsdl:input>
  <soap:body use="literal" />
    </wsdl:input>
- <wsdl:output>
  <soap:body use="literal" />
    </wsdl:output>
    </wsdl:operation>
- <wsdl:operation name="GetNumAutorizacion">
  <soap:operation soapAction="http://tempuri.org/WSMapfre/Service/GetNumAutorizacion"
      style="document" />
- <wsdl:input>
  <soap:body use="literal" />
    </wsdl:input>
- <wsdl:output>
  <soap:body use="literal" />
```

```
    </wsdl:output>
    </wsdl:operation>
- <wsdl:operation name="GetFoto">
  <soap:operation soapAction="http://tempuri.org/WSMapfre/Service/GetFoto" style="document"
    />
- <wsdl:input>
  <soap:body use="literal" />
    </wsdl:input>
- <wsdl:output>
  <soap:body use="literal" />
    </wsdl:output>
    </wsdl:operation>
- <wsdl:operation name="CreateDimeDoc">
  <soap:operation soapAction="http://tempuri.org/WSMapfre/Service/CreateDimeDoc"
    style="document" />
- <wsdl:input>
  <soap:body use="literal" />
    </wsdl:input>
- <wsdl:output>
  <soap:body use="literal" />
    </wsdl:output>
    </wsdl:operation>
    </wsdl:binding>
- <wsdl:binding name="ServiceSoap12" type="tns:ServiceSoap">
  <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />
- <wsdl:operation name="GetConsutaxNombre">
  <soap12:operation soapAction="http://tempuri.org/WSMapfre/Service/GetConsutaxNombre"
    style="document" />
- <wsdl:input>
  <soap12:body use="literal" />
    </wsdl:input>
- <wsdl:output>
  <soap12:body use="literal" />
    </wsdl:output>
    </wsdl:operation>
- <wsdl:operation name="GetConsultaxCodigo">
  <soap12:operation soapAction="http://tempuri.org/WSMapfre/Service/GetConsultaxCodigo"
    style="document" />
- <wsdl:input>
  <soap12:body use="literal" />
    </wsdl:input>
- <wsdl:output>
  <soap12:body use="literal" />
    </wsdl:output>
    </wsdl:operation>
- <wsdl:operation name="GetDatosOtros">
  <soap12:operation soapAction="http://tempuri.org/WSMapfre/Service/GetDatosOtros"
    style="document" />
- <wsdl:input>
  <soap12:body use="literal" />
    </wsdl:input>
- <wsdl:output>
  <soap12:body use="literal" />
    </wsdl:output>
    </wsdl:operation>
- <wsdl:operation name="GetNumAutorizacion">
  <soap12:operation soapAction="http://tempuri.org/WSMapfre/Service/GetNumAutorizacion"
    style="document" />
- <wsdl:input>
  <soap12:body use="literal" />
```

```
      </wsdl:input>
- <wsdl:output>
  <soap12:body use="literal" />
      </wsdl:output>
      </wsdl:operation>
- <wsdl:operation name="GetFoto">
  <soap12:operation soapAction="http://tempuri.org/WSMapfre/Service/GetFoto"
       style="document" />
- <wsdl:input>
  <soap12:body use="literal" />
      </wsdl:input>
- <wsdl:output>
  <soap12:body use="literal" />
      </wsdl:output>
      </wsdl:operation>
      </wsdl:binding>
- <wsdl:service name="Service">
- <wsdl:port name="ServiceSoap" binding="tns:ServiceSoap">
  <soap:address location="http://www2.mapfreperu.com/wsmapfre/wsmapfre.asmx" />
      </wsdl:port>
- <wsdl:port name="ServiceSoap12" binding="tns:ServiceSoap12">
  <soap12:address location="http://www2.mapfreperu.com/wsmapfre/wsmapfre.asmx" />
      </wsdl:port>
      </wsdl:service>
      </wsdl:definitions>
```

ANEXO 05

Ejemplo de Tramas de Respuesta

Parámetros de Configuración:

Código EPS : 060027C (Mapfre)

Código EV : 980008C (Administradora Clínica Ricardo Palma)

RUC EV : 20100121809

5.1.- Operación 1: GetConsultaxnombre

Apellido Paterno	VILCHEZ
Apellido Materno	ORE
Nombre	<<vació>>

En la siguiente trama se muestra los asegurados que coinciden con el Apellido Paterno 'VILCHEZ' y Apellido Materno 'ORE', donde los primeros 6 caracteres corresponden al numero de coincidencias.

<tramaXML>000003VILCHEZ VIVANCO*MAGALY *ORE *6000557*1*S*19870131*F*18* VILCHEZ VIVANCO *BRENDA*ORE*6000558*1*S*19841208*F*18*VILCHEZ VIVANCO*JESSICA*ORE *6000695*1*S*19910810*F*16</tramaXML>

5.2.- Operación 2: GetConsultaxcodigo

Código del Asegurado	06000557
Plan de Salud	18

En la siguiente trama se el asegurados de Código 06000557 y su numero de Plan de Salud 18. Aquí se muestra en la primer parte de la

trama los datos generales del asegurado y en la segunda parte las coberturas del asegurado en dicha clínica.

<tramaXML>00006000557*20108966*VILCHEZ VIVANCO*ORE
*MAGALY*19870131*F*74*44049820
*0006000557*1*18*2*X*20080101*20081231*MAPFRE PER
*20418896915*VILCHEZ VIVANCO*CAHUAS BORJA*CESAR
*2975*PEN*20080101***0*00000000*3 *1*18**ZU*1* * *
*68*80~47 *1*18**Z3*1* * * *0*85~51 *1*18**ZU*0* * * *0*100~52 *1*18**ZU*0*
* * *0*100~BJ *1*18**ZU*0* * * *0*100~Z4 *1*18**ZU*1* * *
*68*80~Z5 *1*18**Z3*1* * **0*85~</tramaXML>